

***Rockchip*  
RK3288  
*Technical Reference Manual***

Revision 0.2  
Mar. 2014

## **Revision History**

<b>Date</b>	<b>Revision</b>	<b>Description</b>
2014-02-25	0.1	Initial Release
2014-03-18	0.2	Add some chapters

## **Warranty Disclaimer**

Rockchip Electronics Co., Ltd makes no warranty, representation or guarantee (expressed, implied, statutory, or otherwise) by or with respect to anything in this document, and shall not be liable for any implied warranties of non-infringement, merchantability or fitness for a particular purpose or for any indirect, special or consequential damages.

Information furnished is believed to be accurate and reliable. However, Rockchip Electronics Co.,Ltd assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use.

Rockchip Electronics Co., Ltd's products are not designed, intended, or authorized for using as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Rockchip Electronics Co., Ltd's product could create a situation where personal injury or death may occur, should buyer purchase or use Rockchip Electronics Co., Ltd's products for any such unintended or unauthorized application, buyers shall indemnify and hold Rockchip Electronics Co., Ltd and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Rockchip Electronics Co., Ltd was negligent regarding the design or manufacture of the part.

## **Copyright and Patent Right**

Information in this document is provided solely to enable system and software implementers to use Rockchip Electronics Co.,Ltd 's products. There are no expressed or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

**Rockchip Electronics Co., Ltd does not convey any license under its patent rights nor the rights of others.**

## **Trademarks**

Rockchip and Rockchip™ logo and the name of Rockchip Electronics Co., Ltd's products are trademarks of Rockchip Electronics Co., Ltd. and are exclusively owned by Rockchip Electronics Co., Ltd. References to other companies and their products use trademarks owned by the respective companies and are for reference purpose only.

## **Confidentiality**

The information contained herein (including any attachments) is confidential. The recipient hereby acknowledges the confidentiality of this document, and except for the specific purpose, this document shall not be disclosed to any third party.

## **Reverse engineering or disassembly is prohibited.**

**ROCKCHIP ELECTRONICS CO.,LTD. RESERVES THE RIGHT TO MAKE CHANGES IN ITS PRODUCTS OR PRODUCT SPECIFICATIONS WITH THE INTENT TO IMPROVE FUNCTION OR DESIGN AT ANY TIME AND WITHOUT NOTICE AND IS NOT REQUIRED TO UNDATE THIS DOCUMENTATION TO REFLECT SUCH CHANGES.**

## **Copyright © 2012 Rockchip Electronics Co., Ltd.**

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electric or mechanical, by photocopying, recording, or otherwise, without the prior written consent of Rockchip Electronics Co., Ltd.

## Table of Content

Table of Content .....	4
Figure Index .....	12
Table Index.....	18
Chapter 1 Introduction .....	21
1.1 Features .....	21
1.2 Block Diagram .....	36
Chapter 2 System Overview .....	37
Chapter 3 Clock & Reset Unit (CRU) .....	44
3.1 Overview .....	44
3.2 Block Diagram .....	44
3.3 System Clock Solution .....	44
3.4 System Reset Solution .....	49
3.5 Function Description .....	49
3.6 PLL Introduction .....	50
3.7 Register Description .....	51
3.8 Timing Diagram .....	138
3.9 Application Notes .....	138
Chapter 4 Power Management Unit (PMU) .....	143
4.1 Overview .....	143
4.2 Block Diagram .....	144
4.3 Power Switch Timing Requirement .....	145
4.4 Function Description .....	146
4.5 Register Description .....	148
4.6 Timing Diagram .....	184
4.7 Application Notes .....	185
Chapter 5 System Debug .....	188
5.1 Overview .....	188
5.2 Block Diagram .....	189
5.3 Function description .....	189
5.4 Register description .....	191
5.5 Interface description .....	229
Chapter 6 System Security.....	232
6.1 Overview .....	232
6.2 Block Diagram .....	232
6.3 Function Description .....	232
6.4 Register Description .....	236
6.5 Application Notes .....	238
Chapter 7 General Register Files (GRF) .....	241
1.1 Overview .....	241
1.2 Function Description .....	241
1.3 GRF Register Description.....	241
1.4 SGRF Register Description .....	错误!未定义书签。
Chapter 8 Core System .....	384
2.1 Overview .....	384
2.2 Block Diagram .....	384
2.3 Function Description .....	384
Chapter 9 Interconnect.....	386

9.1 Overview .....	386
9.2 Block Diagram .....	386
9.3 Function Description(main interconnect).....	386
9.4 Register Description(main interconnect) .....	394
9.5 Function Description(peri interconnect) .....	402
9.6 Register Description(peri interconnect).....	403
9.7 Application Notes .....	406
Chapter 10 DMA Controller for Bus System (DMAC_BUS) .....	408
10.1 Overview .....	408
10.2 Block Diagram .....	408
10.3 Function Description .....	409
10.4 Register Description.....	410
10.5 Timing Diagram .....	425
10.6 Interface Description .....	425
10.7 Application Notes .....	426
Chapter 11 DMA Controller for Peripheral System (DMAC_PERI) .....	434
11.1 Overview .....	434
11.2 Block Diagram .....	434
11.3 Function Description .....	435
11.4 Register Description.....	435
11.5 Timing Diagram .....	450
11.6 Interface Description .....	450
11.7 Application Notes .....	451
Chapter 12 Generic Interrupt Controller (GIC) .....	452
12.1 Overview .....	452
12.2 Block Diagram .....	452
12.3 Function Description .....	452
Chapter 13 Dynamic Memory Interface (DMC) .....	453
13.1 Overview .....	453
13.2 Block Diagram .....	454
13.3 Function description.....	454
13.4 DDR PHY .....	455
13.5 Register description .....	468
13.6 Interface description.....	642
13.7 Application Notes .....	643
Chapter 14 Mobile Storage Host Controller .....	660
14.1 Overview .....	660
14.2 Block Diagram .....	661
14.3 Function Description .....	662
14.4 Register Description.....	682
14.5 Interface Description .....	718
14.6 Application Notes .....	720
Chapter 15 Embedded SRAM .....	748
15.1 Overview .....	748
15.2 Block Diagram .....	748
15.3 Function Description .....	748
Chapter 16 I2S/PCM Controller (8 channel) .....	750
16.1 Overview .....	750

16.2 Block Diagram .....	751
16.3 Function description.....	751
16.4 Register Description.....	755
16.5 Interface description.....	765
16.6 Application Notes .....	766
Chapter 17 SPDIF transmitter.....	768
17.1 Overview .....	768
17.2 Block Diagram .....	768
17.3 Function description.....	769
17.4 Register description .....	772
17.5 Interface description.....	781
17.6 Application Notes .....	782
Chapter 18 USB OTG 2.0 .....	784
18.1 Overview .....	784
18.2 Block Diagram .....	784
18.3 USB OTG2.0 Controller.....	786
18.4 USB OTG2.0 PHY .....	790
18.5 UART BYPASS FUNCITON.....	792
18.6 Register Description.....	793
18.7 Interface description.....	928
18.8 Application Note.....	929
Chapter 19 USB Host 2.0(0).....	931
Chapter 20 USB Host 2.0(1).....	932
20.1 Overview .....	932
20.2 Block Diagram .....	932
20.3 USB Host2.0 Controller .....	932
20.4 USB Host2.0 PHY .....	932
20.5 Register Description.....	937
20.6 Interface description.....	937
20.7 Application Note.....	937
Chapter 21 USB HSIC.....	938
21.1 Overview .....	938
21.2 Block Diagram .....	938
21.3 USB HSIC Controller.....	938
21.4 USB HSIC PHY .....	940
21.5 Register description .....	943
21.6 Application Note.....	974
Chapter 22 HOST Interface .....	975
Chapter 23 Crypto .....	976
23.1 Overview .....	976
23.2 Block Diagram .....	976
23.3 Register description .....	977
23.4 Application Note.....	1000
Chapter 24 GraphicsProcessing Unit (GPU) .....	1002
24.1 Overview .....	1002
24.2 Block Diagram .....	1003
24.3 Function Description .....	1003
24.4 Register Description.....	1003

24.5 Interface Description .....	1003
24.6 Application Notes .....	1003
Chapter 25 Video encoder & decoder Unit (VCODEC) .....	1005
25.1 Overview .....	1005
25.2 Block Diagram .....	1005
25.3 Function Description .....	1006
25.4 Video frame format .....	1007
25.5 Video Decoder .....	1010
25.6 JPEG Decoder .....	1014
25.7 Image Post-processor .....	1016
25.8 Image Pre-processor .....	1018
25.9 H.264 Encoder .....	1019
25.10 JPEG Encoder .....	1020
25.11 MMU .....	1021
25.12 Register Description .....	1021
25.13 Timing Diagram .....	1030
25.14 Interface Description .....	1031
25.15 Application Notes .....	1032
Chapter 26 HEVC .....	1033
Chapter 27 Visual Output Processor (VOP) .....	1034
27.1 Overview .....	1034
27.2 Block Diagram .....	1037
27.3 Function Description .....	1037
27.4 Register Description .....	1062
27.5 Timing Diagram .....	1128
27.6 Interface Description .....	1133
27.7 Application Notes .....	1135
Chapter 28 RGA2 .....	1144
28.1 Overview .....	1144
28.2 Block Diagram .....	1145
28.3 Function Description .....	1146
28.4 Register description .....	1150
28.5 Programming Guide .....	1171
Chapter 29 Image Enhancement Processor (IEP) .....	1173
29.1 Overview .....	1173
29.2 Block Diagram .....	1174
29.3 Function description .....	1175
29.4 Register description .....	1176
29.5 Application Notes .....	1214
Chapter 30 Video Input Processor (VIP) .....	1215
30.1 Overview .....	1215
30.2 Block Diagram .....	1215
30.3 Function description .....	1215
30.4 Register description .....	1218
30.5 Interface description .....	1226
30.6 Application Notes .....	1227
Chapter 31 Image Signal Processing (ISP) .....	1228
31.1 Overview .....	1228

31.2 Block Diagram .....	1229
31.3 Function Description .....	1230
31.4 Register Description.....	1241
31.5 Interface Description .....	1242
31.6 Application Notes .....	1243
Chapter 32 HDMI TX .....	1244
32.1 Overview .....	1244
32.2 Block Diagram .....	1244
32.3 Function Description .....	1244
32.4 Register Description.....	1249
32.5 Interface Description .....	1251
32.6 Application Notes .....	1251
Chapter 33 LVDS .....	1257
33.1 Overview .....	1257
33.2 Block Diagram .....	1257
33.3 Function Description .....	1258
33.4 Register Description.....	1263
33.5 Interface Description .....	1271
33.6 Application Notes .....	1271
Chapter 34 eDP TX Controller.....	1272
34.1 Overview .....	1272
34.2 Block Diagram .....	1273
34.3 Function Description .....	1274
34.4 Register Description.....	1274
34.5 Interface Description .....	1274
34.6 Application Notes .....	1275
Chapter 35 MIPI-CSI PHY .....	1276
35.1 Overview .....	1276
35.2 Block Diagram .....	1276
35.3 Function Description .....	1277
35.4 Register Description.....	1286
35.5 Application Notes .....	1286
Chapter 36 MIPI CSI-2 Host Controller .....	1292
36.1 Overview .....	1292
36.2 Block Diagram .....	1292
36.3 Function Description .....	1293
36.4 Register Description.....	1294
36.5 Application Notes .....	1303
Chapter 37 MIPI Controller.....	1304
37.1 Overview .....	1304
37.2 Block Diagram .....	1304
37.3 Function Description .....	1305
37.4 Register Description.....	1312
37.5 Application Notes .....	1347
Chapter 38 Global Positioning System (GPS).....	1350
38.1 Overview .....	1350
38.2 Block Diagram .....	1350
38.3 Register Summary.....	1350

38.4 Interface Description .....	1352
38.5 Application note .....	1353
Chapter 39 TSP(Transport Stream Processing Module) .....	1354
39.1 Overview .....	1354
39.2 Block Diagram .....	1354
39.3 Function Description .....	1355
39.4 Register Description.....	1358
39.5 Interface Description .....	1399
39.6 Application Notes .....	1400
Chapter 40 High-Speed ADC Interface (HS-ADC).....	1406
40.1 Overview .....	1406
40.2 Block Diagram .....	1406
40.3 Function Description .....	1406
40.4 Register Description.....	1407
40.5 Interface Description .....	1411
40.6 Application Notes .....	1412
Chapter 41 GMAC Ethernet Interface .....	1414
41.1 Overview .....	1414
41.2 Block Diagram .....	1415
41.3 Function Description .....	1415
41.4 Register description .....	1420
41.5 Interface Description .....	1483
41.6 Application Notes .....	1484
Chapter 42 Serial Peripheral Interface (SPI).....	1498
42.1 Overview .....	1498
42.2 Block Diagram .....	1498
42.3 Function description.....	1500
42.4 Register Description.....	1501
42.5 Interface description.....	1513
42.6 Application Notes .....	1513
Chapter 43 PS/2 Controller .....	1516
43.1 Overview .....	1516
43.2 Block Diagram .....	1516
43.3 Function description.....	1517
43.4 Register Description.....	1518
43.5 Interface description.....	1526
43.6 Application Notes .....	1526
Chapter 44 Smart Card Controller.....	1529
44.1 Overview .....	1529
44.2 Block Diagram .....	1529
44.3 Function Description .....	1530
44.4 Register description .....	1534
44.5 Interface Description .....	1549
44.6 Application Notes .....	1549
Chapter 45 SAR-ADC.....	1551
45.1 Overview .....	1551
45.2 Block Diagram .....	1551
45.3 Function description.....	1551

45.4 Register Description.....	1551
45.5 Timing Diagram .....	1553
45.6 Application Notes .....	1554
Chapter 46 Temperature-Sensor ADC(TS-ADC) .....	1555
46.1 Overview .....	1555
46.2 Block Diagram .....	1555
46.3 Function Description .....	1555
46.4 Register Description.....	1556
46.5 Application Notes .....	1564
Chapter 47 Timer .....	1568
47.1 Overview .....	1568
47.2 Block Diagram .....	1568
47.3 Function description.....	1568
47.4 Register Description.....	1569
47.5 Application Notes .....	1571
Chapter 48 eFuse .....	1572
48.1 Overview .....	1572
48.2 Block Diagram .....	1572
48.3 Function description.....	1573
48.4 Register Description.....	1573
48.5 Timing Diagram .....	1574
48.6 Application Notes .....	1576
Chapter 49 General-Purpose Input/Output Ports (GPIO) .....	1577
49.1 Overview .....	1577
49.2 Block Diagram .....	1577
49.3 Function description.....	1577
49.4 Register Description.....	1579
49.5 Interface description.....	1583
49.6 Application Notes .....	1584
Chapter 50 WatchDog .....	1586
50.1 Overview .....	1586
50.2 Block Diagram .....	1586
50.3 Function description.....	1586
50.4 Register Description.....	1588
50.5 Application Notes .....	1591
Chapter 51 Pulse Width Modulation (PWM) .....	1592
45.1 Overview .....	1592
45.2 Block Diagram .....	1593
45.3 Functional description .....	1593
45.4 Register description .....	1594
45.5 Interface Description .....	1610
45.6 Application Notes .....	1610
Chapter 52 I2C Interface .....	1612
52.1 Overview .....	1612
52.2 Block Diagram .....	1612
52.3 Function description.....	1612
52.4 Register Description.....	1615
52.5 Interface description.....	1622

52.6 Application Notes .....	1622
Chapter 53 Universal Asynchronous Receiver/Transmitter (UART) .....	1626
53.1 Overview .....	1626
53.2 Block Diagram .....	1626
53.3 Function description.....	1627
53.4 Register Description.....	1630
53.5 Interface description.....	1648
53.6 Application Notes .....	1648
Chapter 54 Process-Voltage-Temperature Monitor (PVTM) .....	1652
54.1 Overview .....	1652
54.2 Block Diagram .....	1652
Chapter 55 Memory-Management-Unit (MMU) .....	1653
55.1 Overview .....	1653
55.2 Block Diagram .....	1653
55.3 Register Description.....	1656
55.4 MMU Base address .....	1659

## Figure Index

Fig. 1-1 RK3288 Block Diagram .....	36
Fig. 2-1 RK3288 Address Mapping.....	37
Fig. 2-2 RK3288 boot procedure flow.....	39
Fig. 3-1 CRU Architecture .....	44
Fig. 3-2 CRU Clock Architecture Diagram 1.....	45
Fig. 3-3 CRU Clock Architecture Diagram 2.....	46
Fig. 3-4 CRU Clock Architecture Diagram 3.....	47
Fig. 3-5 CRU Clock Architecture Diagram 4.....	48
Fig. 3-6 Reset Architecture Diagram .....	49
Fig. 3-7 PLL Block Diagram .....	50
Fig. 3-8 Chip Power On Reset Timing Diagram .....	138
Fig. 3-9 PLL setting change timing .....	140
Fig. 4-1 Power Domain Partition.....	144
Fig. 4-2 PMU Bock Diagram.....	145
Fig. 4-3 Each Domain Power Switch Timing .....	184
Fig. 4-4 External Wakeup Source PAD Timing .....	185
Fig. 5-1 RK3288 Debug system structure .....	189
Fig. 5-2 Trace funnel architecture.....	190
Fig. 5-3 DAP SWJ interface .....	230
Fig. 6-1 RK3288 security architecture.....	232
Fig. 6-2 TZPC block diagram .....	233
Fig. 6-3 TZMA block diagram.....	233
Fig. 6-4 DMAC_BUS interface .....	234
Fig. 6-5 Software Diagram of Secure and Non-secure .....	238
Fig. 6-6 Embedded SRAM secure memory space setting .....	239
Fig. 8-1 Block Diagram .....	384
Fig. 9-1 Block Diagram .....	386
Fig. 9-2 DDR interleaved example.....	394
Fig. 9-3 Idle request .....	406
Fig. 9-4 DDR timing example .....	407
Fig. 10-1 Block diagram of DMAC_BUS .....	409
Fig. 10-2 DMAC_BUS operation states .....	410
Fig. 10-3 DMAC_BUS request and acknowledge timing.....	425
Fig. 11-1 Block diagram of DMAC_PERI.....	435
Fig. 12-1 Block Diagram .....	452
Fig. 13-1 Protocol controller architecture.....	454
Fig. 13-2 PHY controller architecture .....	454
Fig. 13-3 Protocol controller architecture.....	455
Fig. 13-4 DDR PHY architecture .....	456
Fig. 13-5 DDR PHY master DLL architecture diagram .....	459
Fig. 13-6 DDR PHY master-slave DLL architecture diagram .....	462
Fig. 13-7 Strobe Gating Requirements During Read Operations .....	466
Fig. 13-8 DQS gating – passive windowing mode .....	466
Fig. 13-9 DQS gating – active windowing mode.....	467
Fig. 13-10 Protocol controller architecture .....	645
Fig. 13-11 DLL reset requirements .....	648
Fig. 13-12 DLL reset requirements .....	649
Fig. 13-13 Impedance Calibration Circuit.....	651
Fig. 13-14 I/O cell arrangement with retention .....	653
Fig. 13-15 Sequence of Events to Enter and Exit Retention .....	653
Fig. 14-1 Host Controller Block Diagram.....	661
Fig. 14-2 SD/MMC Card-Detect Signal .....	665
Fig. 14-3 Host Controller Command Path State Machine .....	667
Fig. 14-4 Host Controller Data Transmit State Machine.....	669
Fig. 14-5 Host Controller Data Receive State Machine .....	671
Fig. 14-6 Dual-Buffer Descriptor Structure .....	677

Fig. 14-7 Chain Descriptor Structure .....	677
Fig. 14-8 Descriptor Formats for 32-bit AHB Address Bus Width .....	678
Fig. 14-9 SD/MMC Card-Detect and Write-Protect.....	721
Fig. 14-10 SD/MMC Card Termination .....	721
Fig. 14-11 Host Controller Initialization Sequence.....	724
Fig. 14-12 Voltage Switching Command Flow Diagram .....	734
Fig. 14-13 ACMD41 Argument .....	735
Fig. 14-14 ACMD41 Response(R3).....	735
Fig. 14-15 Voltage Switch Normal Scenario .....	736
Fig. 14-16 Voltage Switch Error Scenario .....	737
Fig. 14-17 CASES for eMMC 4.5 START bit.....	739
Fig. 14-18 Clock Generation Unit .....	741
Fig. 14-19 Card Detection Method 2 .....	746
Fig. 14-20 Card Detection Method 4 .....	747
Fig. 15-1 Embedded SRAM block diagram .....	748
Fig. 16-1 I2S/PCM controller (8 channel) Block Diagram.....	751
Fig. 16-2 I2S transmitter-master & receiver-slave condition.....	752
Fig. 16-3 I2S transmitter-slave& receiver-master condition.....	752
Fig. 16-4 I2S normal mode timing format .....	752
Fig. 16-5 I2S left justified mode timing format.....	753
Fig. 16-6 I2S right justified modetiming format.....	753
Fig. 16-7 PCM early modetiming format .....	754
Fig. 16-8 PCM late1 modetiming format .....	754
Fig. 16-9 PCM late2 modetiming format .....	755
Fig. 16-10 PCM late3 modetiming format .....	755
Fig. 16-11 I2S/PCM controller (8 channel) transmit operation flow chart ...	766
Fig. 16-12 I2S/PCM controller (8 channel) receive operation flow chart ....	767
Fig. 17-1 SPDIF transmitter Block Diagram.....	768
Fig. 17-2 SPDIF Frame Format .....	769
Fig. 17-3 SPDIF Sub-frame Format.....	770
Fig. 17-4 SPDIF Channel Coding .....	770
Fig. 17-5 SPDIF Preamble .....	771
Fig. 17-6 Format of Data-burst .....	772
Fig. 17-7 SPDIF transmitter operation flow chart .....	782
Fig. 18-1 USB OTG 2.0 Architecture .....	784
Fig. 18-2 UTMI interface – Transmit timing for a data packet .....	785
Fig. 18-3 UTMI interface – Receive timing for a data packet .....	786
Fig. 18-4 USB OTG2.0 Controller Architecture .....	787
Fig. 18-5 DFIFO single-port synchronous SRAM interface.....	788
Fig. 18-6 USB OTG 2.0 Controller host mode FIFO address mapping .....	789
Fig. 18-7 USB OTG 2.0 Controller device mode FIFO address mapping .....	790
Fig. 18-8 USB OTG 2.0 PHY Architecture .....	791
Fig. 18-9 USB OTG 2.0 PHY power supply and power up sequence .....	791
Fig. 18-10 UART Application .....	793
Fig. 18-11 UART Timing Sequence .....	793
Fig. 18-12 Resume Timing Sequence .....	929
Fig. 18-13 Reset a port when receiving .....	930
Fig. 18-14 Reset a port when transmitting.....	930
Fig. 20-1 USB HOST 2.0 Architecture.....	932
Fig. 20-2 usb phy architecture.....	933
Fig. 21-1 USB HSIC Architecture.....	938
Fig. 21-2 USB HSIC Controller Architecture .....	939
Fig. 21-3 USB HSIC PHY Architecture .....	941
Fig. 21-4 USB HSIC PHY Digital Block Architecture .....	942
Fig. 23-1 Crypto Architecture .....	976
Fig. 24-1 GPU Block Diagram .....	1003
Fig. 25-1 VCODEC block diagram .....	1005

Fig. 25-2 VCODEC YCbCr 4:2:0 planar format.....	1007
Fig. 25-3 VCODEC YCbCr 4:2:0 Semi-planar format.....	1008
Fig. 25-4 VCODEC Tile scan mode .....	1008
Fig. 25-5 VCODEC YCbCr4:2:2 Interleaved format.....	1009
Fig. 25-6 VCODEC AYCbCr 4:4:4 Interleaved format.....	1009
Fig. 25-7 VCODEC RGB 16bpp format.....	1010
Fig. 25-8 Dataflow of HW performs entropy decoding in video decoder ...	1011
Fig. 25-9 Dataflow of SW performs entropy decoding in video decoder ...	1011
Fig. 25-10 The dataflow of JPEG decoder .....	1015
Fig. 25-11 Post-process standalone dataflow .....	1017
Fig. 25-12 Post-process Pipe-line Mode Dataflow.....	1018
Fig. 25-13 Video Encoder Dataflow.....	1020
Fig. 25-14 structure of two-level page table .....	1021
Fig. 25-15 VCODEC clock structure .....	1030
Fig. 25-16 Aclk_vcodec and Hclk_vcodec Architecture .....	1031
Fig. 25-17 The interrupt interface of vcodec .....	1031
Fig. 27-1 VOP Block Diagram.....	1037
Fig. 27-2 RGB data format .....	1037
Fig. 27-3 YUV data format .....	1038
Fig. 27-4 BPP little/big endian data format .....	1038
Fig. 27-5 LCDC Internal DMA.....	1039
Fig. 27-6 VOP Direct Path Interface .....	1039
Fig. 27-7 De-flicker.....	1050
Fig. 27-8 Virtual display.....	1051
Fig. 27-9 X-Mirror and Y-Mirror.....	1052
Fig. 27-10 overlay .....	1053
Fig. 27-11 post scaling timing .....	1054
Fig. 27-12 Transparency Color Key .....	1055
Fig. 27-13 alpha configuration flow .....	1057
Fig. 27-14 Dither down directy .....	1060
Fig. 27-15 frc pattern diagram.....	1061
Fig. 27-16 dsp_out_mode description .....	1062
Fig. 27-17 LCDC RGB interface timing (SDR) .....	1130
Fig. 27-18 LCDC RGB interface timing (DDR).....	1131
Fig. 27-19 LCDC MCU interface (i80) timing .....	1132
Fig. 27-20 LCDC DPI Programming flow .....	1137
Fig. 27-21 LCDC RGB mode Programming flow .....	1139
Fig. 27-22 LCDC RGB mode Programming flow .....	1140
Fig. 27-23 normal mode left-right type display.....	1141
Fig. 27-24 overlap mode left-right type display .....	1142
Fig. 27-25 command mode flow.....	1142
Fig. 28-1 RGA2 Block Diagram.....	1145
Fig. 28-2 RGA2 in SOC .....	1145
Fig. 28-3 RGA Input Data Format.....	1146
Fig. 28-4 RGA Dither effect .....	1147
Fig. 28-5 RGA Gradient Fill.....	1149
Fig. 28-6 HDMI TX Software Main Sequence Diagram .....	1171
Fig. 28-7 RGA command line and command counter .....	1171
Fig. 28-8 RGA command sync generation .....	1172
Fig. 29-1 IEP block diagram .....	1174
Fig. 30-1 VIP block diagram .....	1215
Fig. 30-2 Timing diagram for VIP when vsync low active.....	1216
Fig. 30-3 Timing diagram for VIP when vsync high active .....	1216
Fig. 30-4 Timing diagram for VIP when href high active.....	1216
Fig. 30-5 Timing diagram for VIP when href low active.....	1216
Fig. 30-6 Timing diagram for VIP when Y data first.....	1216
Fig. 30-7 Timing diagram for VIP when U data first.....	1217

Fig. 30-8 CCIR656 timing .....	1217
Fig. 30-9 Raw Data or JPEG Timing .....	1217
Fig. 31-1 ISP Block Diagram.....	1229
Fig. 31-2 Block Diagram of the Resize Module.....	1236
Fig. 31-3 Memory Organization for the Self Picture Path .....	1238
Fig. 31-4 Definition of Memory Buffers.....	1239
Fig. 31-5 Storage Scheme in Planar and Semi-planar Mode .....	1240
Fig. 32-1 HDMI TX Block Diagram .....	1244
Fig. 32-2 HDMI Video Data Processing .....	1245
Fig. 32-3 HDMI Video Processing Timing .....	1245
Fig. 32-4 HDMI Audio Data Processing Diagram .....	1247
Fig. 32-5 HDMI Audio Clock Regeneration Model .....	1248
Fig. 33-1 LVDS Block Diagram .....	1257
Fig. 33-2 LVDS in SoC.....	1258
Fig. 33-3 LVDS output data timing .....	1258
Fig. 33-4 LVDS h_bp timing diagram .....	1262
Fig. 34-1 eDP TX controller Block Diagram .....	1273
Fig. 34-2 eDP in SoC.....	1274
Fig. 35-1 MIPI D-PHY detailed block diagram .....	1276
Fig. 35-2 MIPI D-PHY Initialization from Shutdown to Idle Modes.....	1279
Fig. 35-3 Power-Up Sequence for Slave Operation .....	1283
Fig. 35-4 HS Data Transfer Sequence .....	1284
Fig. 35-5 HS Data Transfer State Diagram.....	1285
Fig. 35-6 Escape Mode Sequences State Diagram.....	1286
Fig. 36-1 MIPI CSI-2 Host Controller architecture .....	1292
Fig. 37-1 MIPI Controller architecture .....	1304
Fig. 37-2 24bpp APB Pixel to Byte Organization.....	1308
Fig. 37-3 18 bpp APB Pixel to Byte Organization.....	1308
Fig. 37-4 16 bpp APB Pixel to Byte Organization.....	1308
Fig. 37-5 12 bpp APB Pixel to Byte Organization.....	1309
Fig. 37-6 8bpp APB Pixel to Byte Organization .....	1309
Fig. 37-7 Command Transmission Periods within the Image Area .....	1309
Fig. 37-8 Location in the Image Area .....	1312
Fig. 38-1 RK3288 GPS block diagram .....	1350
Fig. 39-1 TSP architecture .....	1355
Fig. 39-2 Sync/Valid Serial Mode with Msb-Lsb Bit Ordering .....	1356
Fig. 39-3 Sync/valid Parallel Mode.....	1356
Fig. 39-4 Sync/Burst Parallel Mode .....	1356
Fig. 39-5 Nosync/Valid Parallel Mode .....	1357
Fig. 40-1 HS-ADC Architecture .....	1406
Fig. 40-2 GPS Application Diagram .....	1407
Fig. 40-3 TS Application Diagram .....	1407
Fig. 40-4 Almost empty triggers a DMA request by DMA request mode....	1412
Fig. 40-5 Almost full triggers a DMA request by DMA request mode .....	1412
Fig. 41-1 GMAC architecture .....	1415
Fig. 41-2 MAC Frame structure .....	1415
Fig. 41-3 RMII transmission bit ordering .....	1416
Fig. 41-4 Start of MII and RMII transmission in 100-Mbps mode.....	1416
Fig. 41-5 End of MII and RMII Transmission in 100-Mbps Mode .....	1417
Fig. 41-6 Start of MII and RMII Transmission in 10-Mbps Mode .....	1417
Fig. 41-7 End of MII and RMII Transmission in 10-Mbps Mode .....	1417
Fig. 41-8 RMII receive bit ordering .....	1417
Fig. 41-9 MDIO frame structure .....	1418
Fig. 41-10 Descriptor Ring and Chain Structure.....	1484
Fig. 41-11 Rx/Tx Descriptors definition .....	1485
Fig. 41-12 RMII clock architecture when clock source from CRU .....	1494
Fig. 41-13 RMII clock architecture when clock source from external OSC.	1494

Fig. 41-14 RGMII clock architecture when clock source from CRU .....	1494
Fig. 41-15 Wake-Up Frame Filter Register .....	1495
Fig. 42-1 SPI Controller Block diagram .....	1499
Fig. 42-2 SPI Master and Slave Interconnection .....	1500
Fig. 42-3 SPI Format (SCPH=0 SCPOL=0).....	1501
Fig. 42-4 SPI Format (SCPH=0 SCPOL=1).....	1501
Fig. 42-5 SPI Format (SCPH=1 SCPOL=0).....	1501
Fig. 42-6 SPI Format (SCPH=1 SCPOL=1).....	1501
Fig. 42-7 SPI Master transfer flow diagram.....	1514
Fig. 42-8 SPI Slave transfer flow diagram.....	1515
Fig. 43-1 PS/2 controller architecture .....	1516
Fig. 43-2 PS/2 host receiving timing .....	1517
Fig. 43-3 PS/2 host sending timing .....	1517
Fig. 43-4 Flow chat for PS/2 controller receiving data mode.....	1527
Fig. 43-5 Flow chat for PS/2 controller sending data mode.....	1528
Fig. 43-6 Flow chat for PS/2 controller inhibition mode.....	1528
Fig. 44-1 SCR Block Diagram.....	1530
Fig. 44-2 Activation, Cold Reset and ATR.....	1532
Fig. 44-3 Warm Reset and ATR .....	1533
Fig. 44-4 Deactivation Sequence.....	1533
Fig. 45-1 RK3288 SAR-ADC block diagram.....	1551
Fig. 45-2 SAR-ADC timing diagram in single-sample conversion mode .....	1553
Fig. 47-1 TS-ADC Controller Block Diagram .....	1555
Fig. 47-2 Single-sample conversion .....	1565
Fig. 47-3 Clock Timing Diagram .....	1565
Fig. 47-1 Timers Block Diagram .....	1568
Fig. 47-2 Timer Usage Flow.....	1569
Fig. 47-3 Timing between timer_en and timer_clk .....	1571
Fig. 48-1 RK3288 eFuse block diagram .....	1572
Fig. 48-2 RK3288 efuse timing diagram in program mode.....	1574
Fig. 48-3 RK3288 efuse timing diagram in read mode .....	1575
Fig. 49-1 GPIO block diagram.....	1577
Fig. 49-2 GPIO Interrupt RTL Block Diagram.....	1579
Fig. 50-1 WDT block diagram .....	1586
Fig. 50-2 WDT Operation Flow .....	1588
Fig. 51-1 PWM architecture .....	1593
Fig. 51-2 PWM Reference Mode.....	1593
Fig. 51-3 PWM Left-aligned Output Mode.....	1594
Fig. 51-4 PWM Center-aligned Output Mode.....	1594
Fig. 51-5 PWM Center-aligned Output Mode.....	1594
Fig. 52-1 I2C architecture .....	1612
Fig. 52-2 I2C DATA Validity .....	1615
Fig. 52-3 I2C Start and stop conditions.....	1615
Fig. 52-4 I2C Acknowledge .....	1615
Fig. 52-5 I2C byte transfer.....	1615
Fig. 52-6 I2C Flow chat for transmit only mode .....	1623
Fig. 52-7 I2C Flow chat for receive only mode .....	1624
Fig. 52-8 I2C Flow chat for mix mode .....	1625
Fig. 53-1 UART Architecture .....	1626
Fig. 53-2 UART Serial protocol .....	1627
Fig. 53-3 IrDA 1.0 .....	1628
Fig. 53-4 UART baud rate.....	1628
Fig. 53-5 UART Auto flow control block diagram .....	1629
Fig. 53-6 UART AUTO RTS TIMING .....	1629
Fig. 53-7 UART AUTO CTS TIMING .....	1630
Fig. 53-8 UART none fifo mode .....	1649
Fig. 53-9 UART fifo mode.....	1649

Fig. 53-10 UART clock generation.....	1650
Fig. 53-11 Power Domain Partition .....	1653

Rockchip Confidential

## Table Index

Table 2-1 RK3288 Interrupt connection list .....	40
Table 2-2 RK3288 DMAC_BUS Hardware request connection list .....	42
Table 2-3 RK3288 DMAC_PERI Hardware request connection list .....	43
Table 4-1 RK3288 Power Domain and Voltage Domain Summary .....	144
Table 4-2 Power Switch Timing .....	145
Table 4-3 Low Power State .....	147
Table 4-4 Wakeup Source .....	147
Table 4-5 Power Domain Status Summary in all Work Mode .....	185
Table 5-1 SWJ interface .....	230
Table 5-2 TPIU interface .....	230
Table 6-1 bus components security setting .....	234
Table 6-2 RK3288 secure device setting .....	235
Table 6-3 RK3288 device secure input port setting .....	235
Table 9-1 Master NIU .....	386
Table 9-2 slave NIU.....	387
Table 9-3 Clock and Power domain .....	388
Table 9-4 DDR configuration.....	391
Table 9-5 DDR Stride .....	392
Table 9-6 Service module.....	394
Table 9-7 Service_bus block .....	395
Table 9-8 Service_core block .....	395
Table 9-9 Service_dmac block .....	396
Table 9-10 Service_gpu block .....	396
Table 9-11 Service_hevc block.....	396
Table 9-12 Service_peri block.....	396
Table 9-13 Service_vio block .....	396
Table 10-1 DMAC_BUS Request Mapping Table .....	408
Table 10-2 DMAC Instruction sets .....	432
Table 11-1 DMAC_PERI Request Mapping Table .....	434
Table 13-1 DDR PHYtrim and test MDLL control .....	459
Table 13-2 charge pump current trim in dll_ctrl.....	460
Table 13-3 DLL digital test control in dll_ctrl .....	460
Table 13-4 DLL analog test control in dll_ctrl .....	460
Table 13-5 bias generator trim in dll_ctrl.....	460
Table 13-6 MDLL feedback trim in dll_ctrl .....	461
Table 13-7 MDLL bypass mode frequency range in dll_ctrl .....	461
Table 13-8 fdtrm control bits in dll_ctrl .....	461
Table 13-9 DDR PHYMSDLL control for trim and test .....	462
Table 13-10 MSDLL digital test control in dll_ctrl .....	463
Table 13-11 MSDLL analog test control in dll_ctrl.....	463
Table 13-12 MSDLL lock detector enable in dll_ctrl .....	464
Table 13-13 slave auto_startup bypass in dll_ctrl .....	464
Table 13-14 slave DLL phase trim in dll_ctrl.....	465
Table 13-15 phase selection for dqs gating .....	466
Table 13-16 dynamic strobe drift indicators .....	467
Table 14-1 Bits in Interrupt Status Register .....	663
Table 14-2 Auto-Stop Generation .....	672
Table 14-3 Non-data Transfer Commands and Requirements .....	674
Table 14-4 Bits in IDMAC DES0 Element.....	678
Table 14-5 Bits in IDMAC DES1 Element.....	679
Table 14-6 Bits in IDMAC DES2 Element.....	679
Table 14-7 Bits in IDMAC DES3 Element.....	680
Table 14-8 IOMUX Settings for SDMMC .....	718
Table 14-9 IOMUX Settings for SDIO0.....	718
Table 14-10 IOMUX Settings for SDIO1 .....	719
Table 14-11 IOMUX Settings for eMMC.....	720

Table 14-12 Recommended Usage of use_hold_reg.....	722
Table 14-13 Command Settings for No-Data Command .....	726
Table 14-14 Command Setting for Single or Multiple-Block Read.....	728
Table 14-15 Command Settings for Single or Multiple-Block Write.....	729
Table 14-16 PBL and Watermark Levels.....	741
Table 14-17 Configuration for SDMMC Clock Generation .....	741
Table 14-18 Configuration for SDIO0 Clock Generation.....	742
Table 14-19 Configuration for SDIO1 Clock Generation.....	743
Table 14-20 Configuration for eMMC Clock Generation.....	743
Table 14-21 Register for SDMMC Card Detection Method 3.....	746
Table 17-1 IOMUX Setting.....	781
Table 18-1 USB OTG 2.0 PHY power supply timing parameter.....	791
Table 18-2 USB OTG 2.0 Interface Description .....	928
Table 20-1 USB HOST 2.0 Interface Description .....	937
Table 25-1 Decoder supported standards, profiles and levels.....	1006
Table 25-2 Encoder supported standard, profile and level .....	1006
Table 25-3 Video decoder H.264 feature.....	1010
Table 25-4 MPEG-4/H.263/SORENSEN SPAR feature.....	1012
Table 25-5 MPEG-2/MPEG-1 features .....	1012
Table 25-6 VC-1 features .....	1013
Table 25-7 RV features .....	1013
Table 25-8 VP6/VP8 features .....	1013
Table 25-9 AVS features .....	1014
Table 25-10 Divx features .....	1014
Table 25-11 JPEG features .....	1014
Table 25-12 Post-processor features.....	1016
Table 25-13 Requirements for post-processor .....	1018
Table 25-14 Post-processor features.....	1018
Table 25-15 Video encoder H.264 feature.....	1019
Table 25-16 JPGE features .....	1020
Table 27-1 alpha blending mode settings .....	1056
Table 27-2 LCDC0 RGB interface(SDR) signal timing constant .....	1130
Table 27-3 LCDC0 RGB interface (DDR) signal timing constant .....	1131
Table 27-4 LCDC1 RGB interface signal timing constant.....	1132
Table 27-5 LCDC0 RGB interface signal timing constant.....	1132
Table 27-6 LCDC1 RGB interface signal timing constant.....	1133
Table 27-7 Gather configuration for all format.....	1136
Table 27-8 effective immediately register table .....	1142
Table 28-1 RGA ROP Boolean operations .....	1150
Table 31-1 ISP Interface Description.....	1242
Table 32-1 HDMI Supported Input Video Formats .....	1245
Table 32-2 HDMI TX I2S 2 Channel Audio Sampling Frequency .....	1247
Table 32-3 HDMI TX I2S 8 Channel Audio Sampling Frequency .....	1247
Table 32-4 HDMI SPDIF Sampling Frequency at Each Video Format.....	1247
Table 32-5 HDMI CTS and N table .....	1248
Table 33-1 MSB mapping relationship (single channel mode) .....	1258
Table 33-2 MSB mapping relationship (double channel mode) .....	1259
Table 34-1 Brief function description of each module in top level .....	1273
Table 35-1 Register Config For D-PHY Mode Select.....	1278
Table 35-2 Frequency Ranges .....	1280
Table 35-3 Power-Up Sequence Timings .....	1283
Table 35-4 Possible Escape Mode Sequences for Data Lanes .....	1285
Table 35-5 DC Specifications .....	1287
Table 35-6 Switching Characteristics.....	1288
Table 35-7 AC Specifications .....	1289
Table 36-1 Supported Camera Settings .....	1293
Table 36-2 Errors Identified by the CSI-2 Host Controller .....	1293

Table 37-1 Color table .....	1305
Table 39-1 TSP Interface Description .....	1399
Table 40-1 IOMUX configuration in TS mode .....	1411
Table 40-2 IOMUX configuration in GPS mode .....	1411
Table 41-1 RMII Interface Description .....	1483
Table 41-2 RGMII Interface Description .....	1483
Table 41-3 Receive Descriptor 0 .....	1485
Table 41-4 Receive Descriptor 1 .....	1487
Table 41-5 Receive Descriptor 2 .....	1487
Table 41-6 Receive Descriptor 3 .....	1488
Table 41-7 Transmit Descriptor 0 .....	1488
Table 41-8 Transmit Descriptor 1 .....	1490
Table 41-9 Transmit Descriptor 2 .....	1491
Table 41-10 Transmit Descriptor 3 .....	1491
Table 42-1 SPI interface description .....	1513
Table 43-1 PS/2 controller Interface Description .....	1526
Table 44-1 IOMUX Setting .....	1549
Table 44-2 BAUDTUNE register .....	1550
Table 45-1 RK3288 SAR-ADC timing parameters list .....	1553
Table 47-1 Timing parameters .....	1565
Table 47-2 Temperature Code Mapping .....	1566
Table 48-1 RK3288 eFuse timing parameters list .....	1575
Table 49-1 GPIO interface description .....	1583
Table 51-1 PWM Interface Description .....	1610
Table 52-1 I2C Interface Description .....	1622
Table 53-1 UART Interface Description .....	1648
Table 53-2 UART baud rate configuration .....	1650

## Chapter 1 Introduction

RK3288 is a low power, high performance processor for mobile phones, personal mobile internet device and other digital multimedia applications, and integrates quad-core Cortex-A17 with separately NEON coprocessor.

Many embedded powerful hardware engines provide optimized performance for high-end application. RK3288 supports almost full-format H.264 decoder by 2160p@24fps, H.265 decoder by 2160p@60fps, also support H.264/MVC/VP8 encoder by 1080p@30fps, high-quality JPEG encoder/decoder, special image preprocessor and postprocessor.

Embedded 3D GPU makes RK3288 completely compatible with OpenGL ES1.1/2.0/3.0, OpenCL 1.1 and DirectX 11. Special 2D hardware engine with MMU will maximize display performance and provide very smoothly operation.

RK3288 has high-performance dual channel external memory interface(DDR3/LPDDR2/LPDDR3) capable of sustaining demanding memory bandwidths, also provides a complete set of peripheral interface to support very flexible applications..

### 1.1 Features

#### 1.1.1 MicroProcessor

- Quad-core ARM Cortex-A17 MPCore processor, a high-performance, low-power and cached application processor
- Full implementation of the ARM architecture v7-A instruction set, ARM Neon Advanced SIMD (single instruction, multiple data) support for accelerated media and signal processing computation
- Superscalar, variable length, out-of-order pipeline with dynamic branch prediction, 8-stage pipeline
- Include VFP v3 hardware to support single and double-precision add, subtract, divide, multiply and accumulate, and square root operations
- SCU ensures memory coherency between the four CPUs
- Integrated 32KB L1 instruction cache , 32KB L1 data cache with 4-way set associative
- 1MB unified L2 Cache
- Trustzone technology support
- Full coresight debug solution
  - Debug and trace visibility of whole systems
  - ETM trace support
  - Invasive and non-invasive debug
- Six separate power domains for every core to support internal power switch and externally turn on/off based on different application scenario
  - PD\_A17\_0: 1<sup>st</sup> Cortex-A17 + Neon + FPU + L1 I/D Cache
  - PD\_A17\_1: 2<sup>nd</sup> Cortex-A17 + Neon + FPU + L1 I/D Cache
  - PD\_A17\_2: 3<sup>rd</sup> Cortex-A17 + Neon + FPU + L1 I/D Cache
  - PD\_A17\_3: 4<sup>th</sup> Cortex-A17 + Neon + FPU + L1 I/D Cache
  - PD\_SCU: SCU + L2 Cache controller, and including PD\_A17\_0, PD\_A17\_1, PD\_A17\_2, PD\_A17\_3, debug logic
- One isolated voltage domain to support DVFS
- Maximum frequency can be up to 1GHz@1.0V

#### 1.1.2 Memory Organization

- Internal on-chip memory
  - 20KB BootRom
  - 100KB internal SRAM for security and non-security access, detailed size is programmable

- External off-chip memory<sup>④</sup>
  - Dual channel DDR3-1066/DDR3L-1066, each channel 16/32bits data widths, 2 ranks, totally 4GB(max) address space, maximum address space for one rank of channel 0 is also 4GB.
  - Dual channel LPDDR2-1066, each channel 32bits data width, 2 ranks, totally 8GB(max) address space, maximum address space for one rank of channel 0 is also 4GB.
  - Dual channel LPDDR3-1066, each channel 32bits data width, 2 ranks, totally 8GB(max) address space, maximum address space for one rank of channel 0 is also 4GB.
  - Dual channel async Nand Flash(include LBA Nand), 8bits data width, 4 banks, 60bits ECC
  - Single channel async Nand Flash(include LBA Nand), 16bits data width, 4 banks, 60bits ECC
  - Dual channel sync ONFI/toggle Nand Flash , 8bits data width, 4 banks, 60bits ECC

### **1.1.3 Internal Memory**

- Internal BootRom
  - Size : 20KB
  - Support system boot from the following device :
    - ◆ 8bits Async Nand Flash
    - ◆ 8bits toggle Nand Flash
    - ◆ SPI interface
    - ◆ eMMC interface
    - ◆ SDMMC interface
  - Support system code download by the following interface:
    - ◆ USB OTG interface
- Internal SRAM
  - Size : 100KB
  - Support security and non-security access
  - Security or non-security space is software programmable
  - Security space can be 0KB,4KB,8KB,12KB,16KB, ... up to 96KB by 4KB step

### **1.1.4 External Memory or Storage device**

- Dynamic Memory Interface (DDR3/DDR3L/LPDDR2/LPDDR3)
  - Compatible with JEDEC standard DDR3/DDR3L/LPDDR2/LPDDR3 SDRAM
  - Data rates up to 1333Mbps(66777MHz) for DDR3/DDR3L
  - Data rates up to 1066Mbps(533MHz) for LPDDR2/LPDDR3
  - Support 2 channel, each channel 16 or 32bits data widths
  - Support up to 2 ranks (chip selects) for each channel, totally 8GB(max) address space, maximum address space for one rank of channel 0 is also 4GB, which is software-configurable.
  - 16bits/32bits data width is software programmable
  - 7 host ports with 64bits/128bits AXI bus interface for system access, AXI bus clock is asynchronous with DDR clock
  - Programmable timing parameters to support DDR3/DDR3L/LPDDR2/LPDDR3 SDRAM from various vendor
  - Advanced command reordering and scheduling to maximize bus utilization
  - Low power modes, such as power-down and self-refresh for DDR3/LPDDR2/LPDDR3 SDRAM; clock stop and deep power-down for LPDDR2 SDRAM
  - Embedded dynamic drift detection in the PHY to get dynamic drift compensation with the controller
  - Programmable output and ODT impedance with dynamic PVT compensation
  - Support one low-power work mode: power down DDR PHY and most of DDR IO except two cs and cke output signals , make SDRAM still in self-refresh state to prevent data missing.

- Nand Flash Interface
  - Support dual channel async nand flash, each channel 8bits, up to 4 banks
  - Support dual channel sync DDR nand flash, each channel 8bits, up to 4 banks
  - Support LBA nand flash in async or sync mode
  - Up to 60bits hardware ECC
  - For DDR nand flash, support DLL bypass and 1/4 or 1/8 clock adjust, maximum clock rate is 75MHz
  - For async nand flash, support configurable interface timing , maximum data rate is 16bit/cycle
  - Embedded special DMA interface to do data transfer
  - Also support data transfer together with general PERI\_DMAMC in SoC system
- eMMC Interface
  - Compatible with standard iNAND interface
  - Support MMC4.5 protocol
  - Provide eMMC boot sequence to receive boot data from external eMMC device
  - Support FIFO over-run and under-run prevention by stopping card clock automatically
  - Support CRC generation and error detection
  - Embedded clock frequency division control to provide programmable baud rate
  - Support block size from 1 to 65535Bytes
  - 8bits data bus width
- SD/MMC Interface
  - Compatible with SD3.0, MMC ver4.5
  - Support FIFO over-run and under-run prevention by stopping card clock automatically
  - Support CRC generation and error detection
  - Embedded clock frequency division control to provide programmable baud rate
  - Support block size from 1 to 65535Bytes
  - Data bus width is 4bits

### 1.1.5 System Component

- CRU (clock & reset unit)
  - Support clock gating control for individual components inside RK3288
  - One oscillator with 24MHz clock input and 5 embedded PLLs
  - Up to 2.2GHz clock output for all PLLs
  - Support global soft-reset control for whole SOC, also individual soft-reset for every components
- PMU(power management unit)
  - Multiple configurable work modes to save power by different frequency or automatical clock gating control or power domain on/off control
  - Lots of wakeup sources in different mode
  - 4 separate voltage domains
  - 12 separate power domains, which can be power up/down by software based on different application scenes
- Timer
  - 8 on-chip 64bits Timers in SoC with interrupt-based operation
  - Provide two operation modes: free-running and user-defined count
  - Support timer work state checkable
  - Fixed 24MHz clock input
- PWM
  - Four on-chip PWMs with interrupt-based operation
  - Programmable pre-scaled operation to bus clock and then further scaled
  - Embedded 32-bit timer/counter facility

- Support capture mode
- Support continuous mode or one-shot mode
- Provides reference mode and output various duty-cycle waveform
- WatchDog
  - 32 bits watchdog counter width
  - Counter clock is from apb bus clock
  - Counter counts down from a preset value to 0 to indicate the occurrence of a timeout
  - WDT can perform two types of operations when timeout occurs:
    - ◆ Generate a system reset
    - ◆ First generate an interrupt and if this is not cleared by the service routine by the time a second timeout occurs then generate a system reset
  - Programmable reset pulse length
  - Totally 16 defined-ranges of main timeout period
- Bus Architecture
  - 128bit/64-bit/32-bit multi-layer AXI/AHB/APB composite bus architecture
  - 5 embedded AXI interconnect
    - ◆ CPU interconnect with four 64-bits AXI masters, one 64-bits AXI slaves, one 32-bits AHB master and lots of 32-bits AHB/APB slaves
    - ◆ PERI interconnect with two 64-bits AXI masters, one 64-bits AXI slave, five 32-bits AHB masters and lots of 32-bits AHB/APB slaves
    - ◆ Display interconnect with three 128-bits AXI master, four 64-bits AXI masters and one 32-bits AHB slave
    - ◆ GPU interconnect with one 128-bits AXI master with point-to-point AXI-lite architecture and 32-bits APB slave
    - ◆ VCODEC interconnect also with two 64-bits AXI master and two 32-bits AHB slave, they are point-to-point AXI-lite architecture
  - For each interconnect with AXI/AHB/APB composite bus, clocks for AXI/AHB/APB domains are always synchronous, and different integer ratio is supported for them.
  - Flexible different QoS solution to improve the utility of bus bandwidth
- Interrupt Controller
  - Support 3 PPI interrupt source and 112 SPI interrupt sources input from different components inside RK3288
  - Support 16 software-triggered interrupts
  - Input interrupt level is fixed , only high-level sensitive
  - Two interrupt outputs (nFIQ and nIRQ) separately for each Cortex-A17, both are low-level sensitive
  - Support different interrupt priority for each interrupt source, and they are always software-programmable
- DMAC
  - Micro-code programming based DMA
  - The specific instruction set provides flexibility for programming DMA transfers
  - Linked list DMA function is supported to complete scatter-gather transfer
  - Support internal instruction cache
  - Embedded DMA manager thread
  - Support data transfer types with memory-to-memory, memory-to-peripheral, peripheral-to-memory
  - Signals the occurrence of various DMA events using the interrupt output signals
  - Mapping relationship between each channel and different interrupt outputs is software-programmable
  - Two embedded DMA controller , BUS\_DMAC is for bus system, PERI\_DMAC is for peripheral system
  - BUS\_DMAC features:
    - ◆ 6 channels totally

- ◆ 6 hardware request from peripherals
- ◆ 2 interrupt output
- ◆ Dual APB slave interface for register config, designated as secure and non-secure
- ◆ Support trustzone technology and programmable secure state for each DMA channel
- PERI\_DMAM features:
  - ◆ 7 channels totally
  - ◆ 9 hardware request from peripherals
  - ◆ 2 interrupt output
  - ◆ Not support trustzone technology
- Security system
  - Support trustzone technology for the following components inside RK3288
    - ◆ Cortex-A17, support security and non-security mode, switch by software
    - ◆ BUS\_DMAM, support some dedicated channels work only in security mode
    - ◆ eFuse, only accessed by Cortex-A17 in security mode
    - ◆ Internal memory , part of space is addressed only in security mode, detailed size is software-programmable together with TZMA(trustzone memory adapter) and TZPC(trustzone protection controller)
  - Embedded encryption and decryption engine
    - ◆ Support AES-128/192/256 with ECB, CBC, OFB, CTR, CBC-MAC, CMAC, XCBC-MAC, XTS and CCM modes
    - ◆ Supports the DES (ECB and CBC modes) and TDES (EDE and DED) algorithms
    - ◆ Supports SHA-1, SHA-256 and SHA-512 modes, as well as HMAC
    - ◆ Support all mathematical operations required to implement the PKA supported cryptosystems between 128 bits and 3136 bits in size (in steps of 32 bits)
    - ◆ Support random bits generator from the ring oscillator
    - ◆ Control the AIB interface to the OTP memory and providing an interface for the CPU to access to the non-confidential trusted data
    - ◆ Set the device's security lifecycle state according to the values of various flag words in the OTP memory
    - ◆ Provide an firmware interface for secure boot, secure debug
    - ◆ Provide a security processor sub-system based on an internal 32-bit CPU
  - Support security boot
  - Support security debug

### 1.1.6 Video CODEC

- Shared internal memory and bus interface for video decoder and encoder<sup>②</sup>
- Embedded memory management unit(MMU)
- Video Decoder
  - Real-time video decoder of MPEG-1, MPEG-2, MPEG-4, H.263, H.264, AVS, VC-1, RV, VP6/VP8, Sorenson Spark, MVC
  - Error detection and concealment support for all video formats
  - Output data format is YUV420 semi-planar, and YUV400(monochrome) is also supported for H.264
  - H.264 up to HP level 5.2 : 2160p@24fps (3840x2160)<sup>③</sup>
  - MPEG-4 up to ASP level 5 : 1080p@60fps (1920x1088)
  - MPEG-2 up to MP : 2160p@24fps (3840x2160)
  - MPEG-1 up to MP : 1080p@60fps (1920x1088)
  - H.263 : 576p@60fps (720x576)
  - Sorenson Spark : 1080p@60fps (1920x1088)
  - VC-1 up to AP level 3 : 1080p@30fps (1920x1088)
  - RV8/RV9/RV10 : 1080p@60fps (1920x1088)
  - VP6/VP8 : 2160p@24fps (3840x2160)
  - AVS : 1080p@60fps (1920x1088)
  - MVC : 2160p@24fps (3840x2160)

- For AVS, 4:4:4 sampling not supported
- For H.264, image cropping not supported
- For MPEG-4, GMC(global motion compensation) not supported
- For VC-1, upscaling and range mapping are supported in image post-processor
- For MPEG-4 SP/H.263/Sorenson spark, using a modified H.264 in-loop filter to implement deblocking filter in post-processor unit
- Video Encoder
  - Support video encoder for H.264 (BP@level4.0, MP@level4.0, HP@level4.0), MVC and VP8
  - Only support I and P slices, not B slices
  - Support error resilience based on constrained intra prediction and slices
  - Input data format:
    - ◆ YCbCr 4:2:0 planar
    - ◆ YCbCr 4:2:0 semi-planar
    - ◆ YCbYCr 4:2:2
    - ◆ CbYCrY 4:2:2 interleaved
    - ◆ RGB444 and BGR444
    - ◆ RGB555 and BGR555
    - ◆ RGB565 and BGR565
    - ◆ RGB888 and BRG888
    - ◆ RGB101010 and BRG101010
  - Image size is from 96x96 to 1920x1088(Full HD)
  - Maximum frame rate is up to 30fps@1920x1080<sup>®</sup>
  - Bit rate supported is from 10Kbps to 20Mbps

### 1.1.7 HEVC Decoder

- Main/Main10 HEVC/H.265 decoder of 4k@60FPS
- Support up to 4096x2304 resolution
- Support up to 100Mbps bit rate
- Embedded memory management unit(MMU)
- Stream error detector (28 IDs)
- Internal 128k cache for bandwidth reduction
- Multi-clock domains and auto clock-gating design for power saving

### 1.1.8 JPEG CODEC

- JPEG decoder
  - Input JPEG file : YCbCr 4:0:0, 4:2:0, 4:2:2, 4:4:0, 4:1:1 and 4:4:4 sampling formats
  - Output raw image : YCbCr 4:0:0, 4:2:0, 4:2:2, 4:4:0, 4:1:1 and 4:4:4 semi-planar
  - Decoder size is from 48x48 to 8176x8176(66.8Mpixels)
  - Support JPEG ROI(region of image) decode
  - Maximum data rate<sup>®</sup> is up to 76million pixels per second
  - Embedded memory management unit(MMU)
- JPEG encoder
  - Input raw image :
    - ◆ YCbCr 4:2:0 planar
    - ◆ YCbCr 4:2:0 semi-planar
    - ◆ YCbYCr 4:2:2
    - ◆ CbYCrY 4:2:2 interleaved
    - ◆ RGB444 and BGR444
    - ◆ RGB555 and BGR555
    - ◆ RGB565 and BGR565
    - ◆ RGB888 and BRG888
    - ◆ RGB101010 and BRG101010
  - Output JPEG file : JFIF file format 1.02 or Non-progressive JPEG

- Encoder image size up to 8192x8192(64million pixels) from 96x32
- Maximum data rate<sup>④</sup> up to 90million pixels per second
- Embedded memory management unit(MMU)

### 1.1.9 Image Enhancement

- Image pre-processor
  - Only used together with HD video encoder inside RK3288, not support stand-alone mode
  - Provides RGB to YCbCr 4:2:0 color space conversion, compatible with BT601, BT709 or user defined coefficients
  - Provides YCbCr4:2:2 to YCbCr4:2:0 color space conversion
  - Support cropping operation from 8192x8192 to any supported encoding size
  - Support rotation with 90 or 270 degrees
- Video stabilization
  - Work in combined mode with HD video encoder inside RK3288 and stand-alone mode
  - Adaptive motion compensation filter
  - Support scene detection from video sequence, encodes key frame when scene change noticed
- Image Post-Processor (embedded inside video decoder)
  - Combined with HD video decoder and JPEG decoder, post-processor can read input data directly from decoder output to reduce bus bandwidth
  - Also work as a stand-alone mode, its input data is from image data stored in external memory
  - Input data format:
    - ◆ Any format generated by video decoder in combined mode
    - ◆ YCbCr 4:2:0 semi-planar
    - ◆ YCbCr 4:2:0 planar
    - ◆ YCbYCr 4:2:2
    - ◆ YCrYCb 4:2:2
    - ◆ CbYCrY 4:2:2
    - ◆ CrYCbY 4:2:2
  - Output data format:
    - ◆ YCbCr 4:2:0 semi-planar
    - ◆ YCbYCr 4:2:2
    - ◆ YCrYCb 4:2:2
    - ◆ CbYCrY 4:2:2
    - ◆ CrYCbY 4:2:2
    - ◆ Fully configurable ARGB channel lengths and locations inside 32bits, such as ARGB8888, RGB565, ARGB4444 etc.
  - Input image size:
    - ◆ Combined mode: from 48x48 to 8176x8176 (66.8Mpixels)
    - ◆ Stand-alone mode: width from 48 to 8176, height from 48 to 8176, and maximum size limited to 16.7Mpixels
    - ◆ Step size is 16 pixels
  - Output image size: from 16x16 to 1920x1088 (horizontal step size 8, vertical step size 2)
  - Support image up-scaling:
    - ◆ Bicubic polynomial interpolation with a four-tap horizontal kernel and a two-tap vertical kernel
    - ◆ Arbitrary non-integer scaling ratio separately for both dimensions
    - ◆ Maximum output width is 3x input width
    - ◆ Maximum output height is 3x input height
  - Support image down-scaling:
    - ◆ Arbitrary non-integer scaling ratio separately for both dimensions
    - ◆ Unlimited down-scaling ratio

- Support YUV to RGB color conversion, compatible with BT.601-5, BT.709 and user definable conversion coefficient
- Support dithering (2x2 ordered spatial dithering) for 4/5/6bit RGB channel precision
- Support programmable alpha channel and alpha blending operation with the following overlay input formats:
  - ◆ 8bit alpha + YUV444, big endian channel order with AYUV8888
  - ◆ 8bit alpha + 24bit RGB, big endian channel order with ARGB8888
- Support deinterlacing with conditional spatial deinterlace filtering, only compatible with YUV420 input format
- Support RGB image contrast/brightness/color saturation adjustment
- Support image cropping & digital zoom only for JPEG or stand-alone mode
- Support picture in picture
- Support image rotation (horizontal flip, vertical flip, rotation 90,180 or 270 degrees)
- Image Enhancement-Processor (IEP)
  - Image format
    - ◆ Input data: XRGB/RGB565/YUV420/YUV422
    - ◆ Output data: ARGB/RGB565/YUV420/YUV422
    - ◆ The format ARGB/XRGB/RGB565/YUV support swap
    - ◆ Support YUV semi-planar/planar
    - ◆ Support BT601\_I/BT601\_f/BT709\_I/BT709\_f color space conversion
    - ◆ Support RGB dither up/down conversion
    - ◆ Support YUV up/down sampling conversion
    - ◆ Max source image resolution: 8192x8192
    - ◆ Max scaled image resolution: 4096x4096
  - Enhancement
    - ◆ Gamma adjustment with programmable mapping table
    - ◆ Hue/Saturation/Brightness/Contrast enhancement
    - ◆ Color enhancement with programmable coefficient
    - ◆ Detail enhancement with filter matrix up to 9x9
    - ◆ Edge enhancement with filter matrix up to 9x9
    - ◆ Programmable difference table for detail enhancement
    - ◆ Programmable distance table for detail and edge enhancement
  - Noise reduction
    - ◆ Compression noise reduction with filter matrix up to 9x9
    - ◆ Programmable difference table for compression noise reduction
    - ◆ Programmable distance table for compression noise reduction
    - ◆ Spatial sampling noise reduction
    - ◆ Temporal sampling noise reduction
    - ◆ Optional coefficient for sampling noise reduction
  - Scaling
    - ◆ Horizontal down-scaling with vertical down-scaling
    - ◆ Horizontal down-scaling with vertical up-scaling
    - ◆ Horizontal up-scaling with vertical down-scaling
    - ◆ Horizontal up-scaling with vertical up-scaling
    - ◆ Arbitrary non-integer scaling ratio, from 1/16 to 16
  - Deinterlace
    - ◆ Input 4 fields, output 2 frames mode
    - ◆ Input 4 fields, output 1 frames mode
    - ◆ Input 2 fields, output 1 frames mode
    - ◆ Programmable motion detection coefficient
    - ◆ Programmable high frequency factor
    - ◆ Programmable edge interpolation parameter
    - ◆ Source width up to 1920
  - Interface
    - ◆ Programmable direct path to VOP

- Embedded memory management unit(MMU)

### 1.1.10 Graphics Engine

- 3D Graphics Engine :
  - High performance OpenGL ES1.1/2.0/3.0, OpenCL 1.1, DirectX 11 etc.
  - Embedded 4 shader cores with shared hierarchical tiler
  - Provide MMU and L2 Cache with 256KB size
  - Image quality using double-precision FP64, and anti-aliasing
- 2D Graphics Engine :
  - BitBlit with Stretch Blit, Simple Blit and Filter Blit
  - Color fill with gradient fill, and pattern fill
  - Line drawing with anti-aliasing and specified width
  - High-performance stretch and shrink
  - Monochrome expansion for text rendering
  - ROP2, ROP3, ROP4
  - Alpha blending modes including global alpha, per pixel alpha, porter-duff and fading
  - 8K x 8K input and 2K x 2K output raster 2D coordinate system
  - Arbitrary degrees rotation with anti-aliasing on every 2D primitive
  - Blending, scaling and rotation are supported in one pass for Bitblit
  - Source format:
    - ◆ ABGR8888, XBGR888, ARGB8888, XRGB888
    - ◆ RGB888, RGB565
    - ◆ RGBA5551, RGBA4444
    - ◆ YUV420 planar, YUV420 semi-planar
    - ◆ YUV422 planar, YUV422 semi-planar
    - ◆ BPP8, BPP4, BPP2, BPP1
  - Destination formats:
    - ◆ ABGR8888, XBGR888, ARGB8888, XRGB888
    - ◆ RGB888, RGB565
    - ◆ RGBA5551, RGBA4444
    - ◆ YUV420 planar, YUV420 semi-planar only in filter and pre-scale mode
    - ◆ YUV422 planar, YUV422 semi-planar only in filter and pre-scale mode

### 1.1.11 Video IN/OUT

- Camera Interface(interface only)
  - Support up to 5M pixels
  - 8bits BT656(PAL/NTSC) interface
  - 16bits BT601 DDR interface
  - 8bits/10bits/12bits raw data interface
  - YUV422 data input format with adjustable YUV sequence
  - YUV422,YUV420 output format with separately Y and UV space
  - Support picture in picture (PIP)
  - Support simple image effects such as Arbitrary(sepia), Negative, Art freeze, Embossing etc.
  - Support static histogram statistics and white balance statistics
  - Support image crop with arbitrary windows
  - Support scale up/down from 1/8 to 8 with arbitrary non-integer ratio
- Camera Interface and Image Processor(Interface and Image Processing)
  - Maximum input resolution of 14M(4416x3312) pixels
  - Main scaler with pixel-accurate up- and down-scaling to any resolution between 4416x3312 and 32x16 pixel in processing mode
  - Self scaler with pixel-accurate up- and down-scaling to any resolution between 1920x1080 and 32x16 pixel in processing mode
  - support of semiplanar NV21 color storage format

- support of independent image cropping on main and self path
- ITU-R BT 601/656 compliant video interface supporting YCbCr or RGB Bayer data
- 12 bit camera interface
- 12 bit resolution per color component internally
- YCbCr 4:2:2 processing
- Hardware JPEG encoder incl. JFIF1.02 stream generator and programmable quantization and Huffman tables
- Windowing and frame synchronization
- Frame skip support for video (e.g. MPEG-4) encoding
- Macro block line, frame end, capture error, data loss interrupts and sync. (h\_start, v\_start) interrupts
- Luminance/chrominance and chrominance blue/red swapping for YUV input signals
- Continuous resize support
- Color processing (contrast, saturation, brightness, hue, offset, range)
- Display-ready RGB output in self-picture path (RGB888, RGB666 and RGB565)
- Rotation unit in self-picture path (90°, 180°, 270° and h/v flipping) for RGB output
- Read port provided to read back a picture from system memory
- Simultaneous picture read back, resizing and storing through self path while main path captures the camera picture
- Black level compensation
- Four channel Lens shade correction (Vignetting)
- Auto focus measurement
- White balancing and black level measurement
- Auto exposure support by brightness measurement in 5x5 sub windows
- Defect pixel cluster correction unit (DPCC) supports on the fly and table based pixel correction
- De-noising pre filter (DPF)
- Enhanced color interpolation (RGB Bayer demosaicing)
- Chromatic aberration correction
- Combined edge sensitive Sharpening / Blurring filter (Noise filter)
- Color correction matrix (cross talk matrix)
- Global Tone Mapping with wide dynamic range unit (WDR)
- Image Stabilization support and Video Stabilization Measurement
- Flexible Histogram calculation
- Digital image effects (Emboss, Sketch, Sepia, B/W (Grayscale), Color Selection, Negative image, sharpening)
- Solarize effect through gamma correction
- Display Interface
  - Embedded two channel display interfaces: VOP\_BIG and VOP\_LIT.
  - Parallel Display interface
    - ◆ Parallel RGB LCD Interface:
      - 30-bit(RGB101010), 24-bit(RGB888), 18-bit(RGB666), 15-bit(RGB565)
    - ◆ Serial RGB LCD Interface(optional):
      - 2x12-bit, 3x8-bit(RGB delta support), 3x8-bit+dummy
    - ◆ MCU LCD interface(optional):
      - i-8080(up to 24-bit RGB), Hold/Auto/Bypass modes
    - ◆ TV Interface: ITU-R BT.656(8-bit, 480i/576i/1080i)
    - ◆ DDR output interface:
      - parallel RGB and 2x12-bit serial RGB
      - Single or dual clock out
    - ◆ dither down:
      - allegro, FRC
      - gamma after dither
    - ◆ Max output resolution: 3840x2160 (for VOP\_BIG), 2560x1600 (for VOP\_LIT)
    - ◆ Scanning timing 8192x4096
  - Display process

- ◆ Background layer:
  - programmable 24-bit color
- ◆ Win0 (Video0) layer:
  - RGB888, ARGB888, RGB565, YCbCr422, YCbCr420, YCbCr444
  - Support virtual display
  - 1/8 to 8 scaling-down and scaling-up engine:
    - ✧ Scale up using bicubic or bilinear;
    - ✧ Scale down using bilinear or average;
    - ✧ 4 Bicubic tables : precise,spline,catrom,mitchell;
    - ✧ coord 8bit, coe 8bit signed
  - x-mirror,y-mirror
- ◆ Win1 (Video1) layer:
  - RGB888, ARGB888, RGB565, YCbCr422, YCbCr420, YCbCr444
  - Support virtual display
  - 1/8 to 8 scaling-down and scaling-up engine
    - ✧ Scale up using bicubic or bilinear;
    - ✧ Scale down using bilinear otraverage;
    - ✧ 4 Bicubic tables : precise,spline,catrom,mitchell;
    - ✧ coord 8bit, coe 8bit signed
  - x-mirror,y-mirror
- ◆ Win2 (UI 0) layer:
  - RGB888, ARGB888, RGB565, 1/2/4/8bpp
  - Support virtual display
  - 4 display regions
  - x-mirror,y-mirror
- ◆ Win3 (UI 1) layer:
  - RGB888, ARGB888, RGB565, 1/2/4/8bpp
  - Support virtual display
  - 4 display regions
  - x-mirror,y-mirror
- ◆ Hardware cursor:
  - RGB888, ARGB888, RGB565, 1/2/4/8bpp
  - Support two size: 32x32,64x64,or 128x128
- ◆ Overlay:
  - Win0/Win1/Win2/Win3 256 level alpha blending (support pre-multiplied alpha)
  - Win0/Win1/Win2/Win3 overlay position exchangeable
  - Win0/Win1/Win2/Win3 Transparency color key
  - Win0/Win1/Win2/Win3 global/per-pixel alpha
  - HWC 256 level alpha blending
  - HWC global/per-pixel alpha
- Others
  - ◆ 3 x 256 x 8 bits display LUTs
  - ◆ YcbCr2RGB(rec601-mpeg/rec601-jpeg/rec709/BT2020)and RGB2YcbCr
  - ◆ Support BCSH function
  - ◆ Support CABC function
  - ◆ QoS request signals
  - ◆ Gather transfer (Max 8)
  - ◆ Y/UV scheduler
  - ◆ Addr alignment
  - ◆ Support IEP direct path(win0/1/2/3)
  - ◆ Embedded memory management unit(MMU)
  - ◆ Support MIPI flow control

### 1.1.12 HDMI

- Single Physical Layer PHY with support for HDMI 1.4 and 2.0 operation
- For HDMI operation, support for the following:

- Up to 1080p at 120 Hz and 4k x 2k at 60 Hz HDTV display resolutions and up to QXGA graphic display resolutions
- 3-D video formats
- Up to 10-bit Deep Color modes
- Up to 18 Gbps aggregate bandwidth
- 13.5–600 MHz input reference clock
- HPD input analog comparator
- Link controller flexible interface with 30-, 60- or 120-bit SDR data access
- Support HDCP 1.4

### **1.1.13 LVDS**

- Comply with the TIA/EIA-644-A LVDS standard
- Combine LVTTL IO, support LVDS/LVTTL data output
- Support reference clock frequency range from 10Mhz to 148.5Mhz
- Support LVDS RGB 30/24/18bits color data transfer
- Support VESA/JEIDA LVDS data format transfer
- Support LVDS single channel and double channel data transfer, every channel include 5 data lanes and 1 clock lane

### **1.1.14 MIPI PHY**

- Embedded 3 MIPI PHY, MIPI 0 only for TX, MIPI 1 for TX and RX, MIPI 2 only for RX
- Support 4 data lane, providing up to 6Gbps data rate
- Support 1080p @ 60fps output
- Lane operation ranging from 80 Mbps to 1.5 Gbps in forward direction

### **1.1.15 eDP PHY**

- Support 4Kx2K @ 30fps
- Compliant with eDP TM Specification, version 1.1
- Up to 4 physical lanes of 2.7/1.62 Gbps/lane(HBR2/HBR/RBR)
- RGB, YCbCr 4:4:4, YCbCr 4:2:2 and 8/10/12 bit per component video format
- Encoded bit stream (Dolby Digital, or DTS) – IEC61937 compliant
- Support VESA DMT and CTV timing standards
- Fully support EIA/CEA-861D video timing and Info Frame structure
- Hot plug and unplug detection and link status monitor
- Support DDC/CI and MCCS command transmission when the monitor includes a display controller.
- Supports Panel Self Refresh(PSR)

### **1.1.16 Audio Interface**

- I2S/PCM with 8ch
  - Up to 8 channels (4xTX, 2xRX)
  - Audio resolution from 16bits to 32bits
  - Sample rate up to 192KHz
  - Provides master and slave work mode, software configurable
  - Support 3 I2S formats (normal, left-justified, right-justified)
  - Support 4 PCM formats(early, late1, late2, late3)
  - I2S and PCM mode cannot be used at the same time
- SPDIF
  - Support two 16-bit audio data store together in one 32-bit wide location
  - Support biphase format stereo audio data output
  - Support 16 to 31 bit audio data left or right justified in 32-bit wide sample data buffer
  - Support 16, 20, 24 bits audio data transfer in linear PCM mode
  - Support non-linear PCM transfer

### 1.1.17 Connectivity

- SDIO interface
  - Embedded 2 SDIO interface
  - Compatible with SDIO 3.0 protocol
  - 4bits data bus widths
- High-speed ADC stream interface
  - Support single-channel 8bits/10bits interface
  - DMA-based and interrupt-based operation
  - Support 8bits TS stream interface
- TS interface
  - Supports two TS input channels and one TS output channel.
  - Supports 4 TS Input Mode: sync/valid mode in the case of serial TS input; nosync/valid mode, sync/valid, sync/burst mode in the case of parallel TS input.
  - Supports serial and parallel output mode with PCR adjustment, and lsb-msb or msb-lsb bit ordering can be chosen in the serial output mode.
  - Supports 2 TS sources: demodulators and local memory.
  - Supports 2 Built-in PTIs(Programmable Transport Interface) to process TS simultaneously, and Each PTI supports:
    - ◆ 64 PID filters.
    - ◆ TS descrambling with 16 sets of Control Word under CSA v2.0 standard, up to 104Mbps
    - ◆ 16 PES/ES filters with PTS/DTS extraction and ES start code detection.
    - ◆ 4/8 PCR extraction channels
    - ◆ 64 Section filters with CRC check, and three interrupt mode: stop per unit, full-stop, recycle mode with version number check
    - ◆ PID done and error interrupts for each channel
    - ◆ PCR/DTS/PTS extraction interrupt for each channel
  - Supports 1 PVR(Personal Video Recording) output channel.
  - 1 built-in multi-channel DMA Controller.
- PS2 interface
  - Support PS/2 data communication protocol
  - Support PS/2 master mode
  - Software programmable timing requirement to support max PS/2 clock frequency to 33KHZ
  - Support status to be queried for data communication error
  - Support interrupt mode for data communication finish
  - Support timeout mechanism for data communication
  - Support interrupt mode for data communication timeout
- Smart Card
  - support card activation and deactivation
  - support cold/warm reset
  - support Answer to Reset (ATR) response reception
  - support T0 for asynchronous half-duplex character transmission
  - support T1 for asynchronous half-duplex block transmission
  - support automatic operating voltage class selection
  - support adjustable clock rate and bit (baud) rate
  - support configurable automatic byte repetition
- Host interface
  - Low Pin Count interface(8 inputs/16 outputs or 16 inputs/8 outputs)
  - No mandatory Tri-State signals
  - All signals driven using source synchronous clock.(2 DDR clock signals per direction for TX and RX paths)

- Low latency through serialization/deserialization
- Transport clocks and bus clock are independent
- Support Asymmetric(Host/Peripheral) communication operations
- Support multiple outstanding transactions Reads, Writes and interrupts
- Support Mirror Mode to enable self test with identical device
- GPS Interface
  - Single chip, integrate GPS bb with cpu
  - 32 DMA channels for AHB master access
  - Complete 1-band, C/A, and NMEA-0183 compatibility
  - Support reference frequencies 16.368MHz
  - High sensitivity for indoor fixes
  - Low power consumption
  - Low cost with smaller size
  - Multi modes support both standalone GPS and A\_GPS
- GMAC 10/100/1000M Ethernet Controller
  - Supports 10/100/1000-Mbps data transfer rates with the RGMII interfaces
  - Supports 10/100-Mbps data transfer rates with the RMII interfaces
  - Supports both full-duplex and half-duplex operation
    - ◆ Supports CSMA/CD Protocol for half-duplex operation
    - ◆ Supports packet bursting and frame extension in 1000 Mbps half-duplex operation
    - ◆ Supports IEEE 802.3x flow control for full-duplex operation
    - ◆ Optional forwarding of received pause control frames to the user application in full-duplex operation
    - ◆ Back-pressure support for half-duplex operation
    - ◆ Automatic transmission of zero-quanta pause frame on deassertion of flow control input in full-duplex operation
  - Preamble and start-of-frame data (SFD) insertion in Transmit, and deletion in Receive paths
  - Automatic CRC and pad generation controllable on a per-frame basis
  - Options for Automatic Pad/CRC Stripping on receive frames
  - Programmable Inter Frame Gap (40-96 bit times in steps of 8)
  - Supports a variety of flexible address filtering modes
  - Separate 32-bit status returned for transmission and reception packets
  - Supports IEEE 802.1Q VLAN tag detection for reception frames
  - Support detection of LAN wake-up frames and AMD Magic Packet frames
  - Support checksum off-load for received IPv4 and TCP packets encapsulated by the Ethernet frame
  - Support checking IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagrams
  - Comprehensive status reporting for normal operation and transfers with errors
  - Automatic generation of PAUSE frame control or backpressure signal to the GMAC core based on Receive FIFO-fill (threshold configurable) level
  - Handles automatic retransmission of Collision frames for transmission
  - Discards frames on late collision, excessive collisions, excessive deferral and under run conditions
- SPI Controller
  - 3 on-chip SPI controller inside RK3288
  - Support serial-master and serial-slave mode, software-configurable
  - DMA-based or interrupt-based operation
  - Embedded two 32x16bits FIFO for TX and RX operation respectively
  - Support 2 chip-selects output in serial-master mode
- Uart Controller

- 5 on-chip uart controller inside RK3288
- DMA-based or interrupt-based operation
- For all UART, two 64Bytes FIFOs are embedded for TX/RX operation respectively
- Support 5bit,6bit,7bit,8bit serial data transmit or receive
- Standard asynchronous communication bits such as start,stop and parity
- Support different input clock for uart operation to get up to 4Mbps or other special baud rate
- Support non-integer clock divides for baud clock generation
- Auto flow control mode is for all UART, except UART\_DBG
- I2C controller
  - 6 on-chip I2C controller in RK3288
  - Multi-master I2C operation
  - Support 7bits and 10bits address mode
  - Software programmable clock frequency and transfer rate up to 400Kbit/s in the fast mode
  - Serial 8bits oriented and bidirectional data transfers can be made at up to 100Kbit/s in the standard mode
- GPIO
  - Totally 160 GPIOs
  - All of GPIOs can be used to generate interrupt to Cortex-A17
  - GPIO0 can be used to wakeup system from low-power mode
  - The pull direction(pullup or pulldown) for all of GPIOs are software-programmable
  - All of GPIOs are always in input direction in default after power-on-reset
  - The drive strength for all of GPIOs is software-programmable
- USB Host2.0
  - Embedded 2 USB Host2.0 interfaces
  - Compatible with USB Host2.0 specification
  - Supports high-speed(480Mbps), full-speed(12Mbps) and low-speed (1.5Mbps) mode
  - Provides 16 host mode channels
  - Support periodic out channel in host mode
- USB OTG2.0
  - Compatible with USB OTG2.0 specification
  - Supports high-speed(480Mbps), full-speed(12Mbps) and low-speed (1.5Mbps) mode
  - Support up to 9 device mode endpoints in addition to control endpoint 0
  - Support up to 6 device mode IN endpoints including control endpoint 0
  - Endpoints 1/3/5/7 can be used only as data IN endpoint
  - Endpoints 2/4/6 can be used only as data OUT endpoint
  - Endpoints 8/9 can be used as data OUT and IN endpoint
  - Provides 9 host mode channels
- HSIC Interface
  - Compliant with the USB2.0 Specification and Enhanced Host Controller Interface Specification 2.0
  - 1 Port HSIC PHY Interface Operates in host mode
  - Built-in one 512x64 bits FIFO
  - Internal DMA with scatter/gather function

### **1.1.18 Others**

- Temperature Sensor(TS-ADC)
  - 3 bipolar-based temperature-sensing cell embedded
  - 3-channel 12-bits SAR ADC
  - Temperature accuracy sensed is  $\pm 5$  degree
  - SAR-ADC clock must be less than 50KHz

- Power Down Current is about 1uA for analog and 2uA for digital logic
- SAR-ADC(Successive Approximation Register)
  - 3-channel single-ended 10-bit SAR analog-to-digital converter
  - Conversion speed range is up to 1 MSPS
  - SAR-ADC clock must be less than 1MHz
  - DNL is less than  $\pm 1$  LSB , INL is less than  $\pm 2.0$  LSB
  - Power down current is about 0.5uA for analog and digital logic
  - Power supply is 1.8V ( $\pm 10\%$ ) for analog interface
- eFuse
  - Two high-density electrical Fuse is integrated: 256bits (32x8) / 1024bits (32x32)
  - Support standby mode

*Notes :<sup>①</sup>: DDR3/LPDDR2/LPDDR3 are not used simultaneously as well as async and sync ddr nand flash*

*<sup>②</sup>: In RK3288, Video decoder and encoder are not used simultaneously because of shared internal buffer*

*<sup>③</sup>: Actual maximum frame rate will depend on the clock frequency and system bus performance*

*<sup>④</sup>: Actual maximum data rate will depend on the clock frequency and JPEG compression rate*

## 1.2 Block Diagram

The following diagram shows the basic block diagram for RK3288.

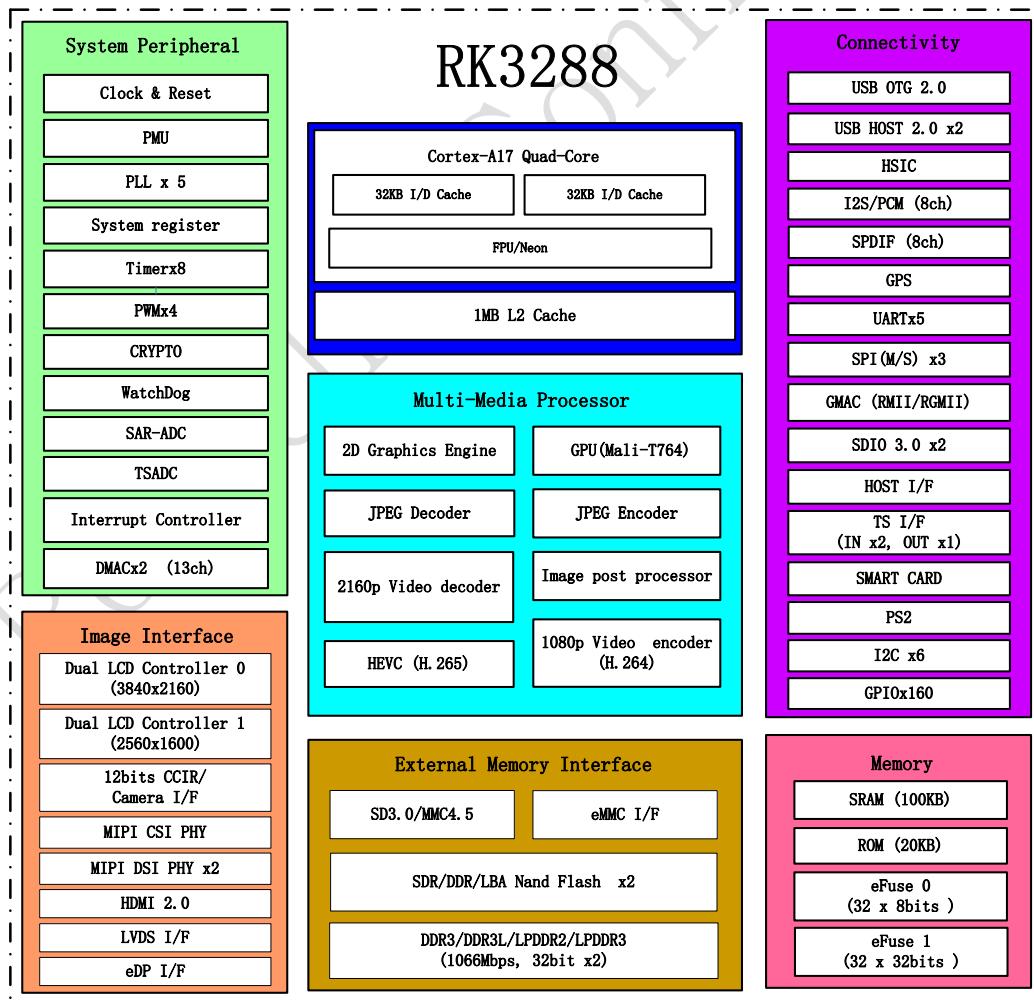


Fig. 1-1 RK3288 Block Diagram

## Chapter 2 System Overview

### 2.1 Address Mapping

RK3288 support to boot from internal bootrom, which support remap function by software programming. Remap is controlled by SGRF\_SOC\_CON0 bit[11].

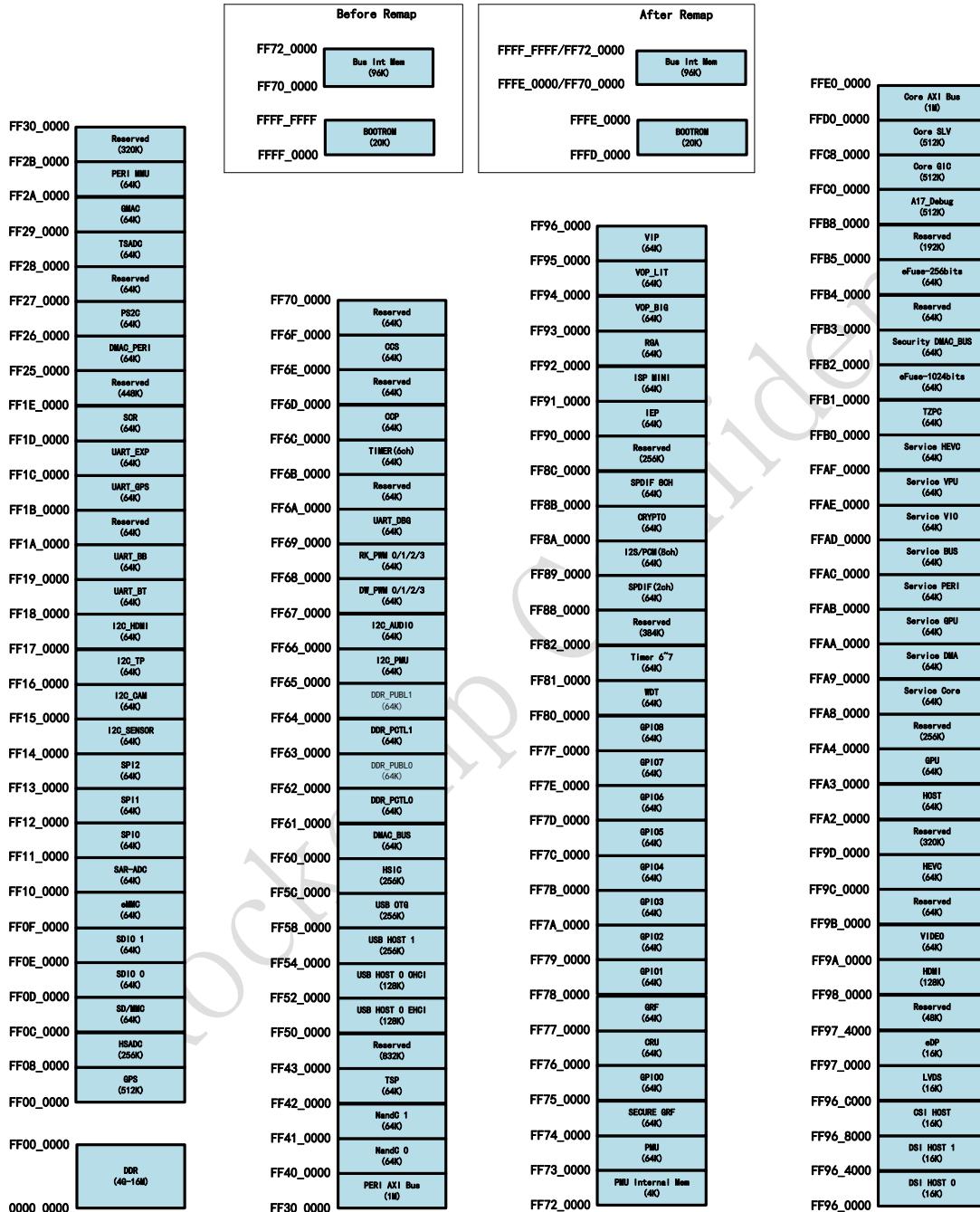


Fig. 2-1 RK3288 Address Mapping

### 2.2 System Boot

RK3288 provides system boot from off-chip devices such as SDMMC card, 8bits async nand flash or toggle nand flash, SPI nor or nand, and eMMC memory. When boot code is not ready

in these devices, also provide system code download into them by USB OTG interface. All of the boot code will be stored in internal bootrom. The following is the whole boot procedure for boot code, which will be stored in bootrom in advance.

The following features are supports.

- Support secure boot mode and non-secure boot mode
- Support system boot from the following device:
  - 8bits Async Nand Flash
  - 8bits Toggle Nand Flash
  - SPI2\_CS0 interface
  - eMMC interface
  - SDMMC Card
- Support system code download by USB OTG

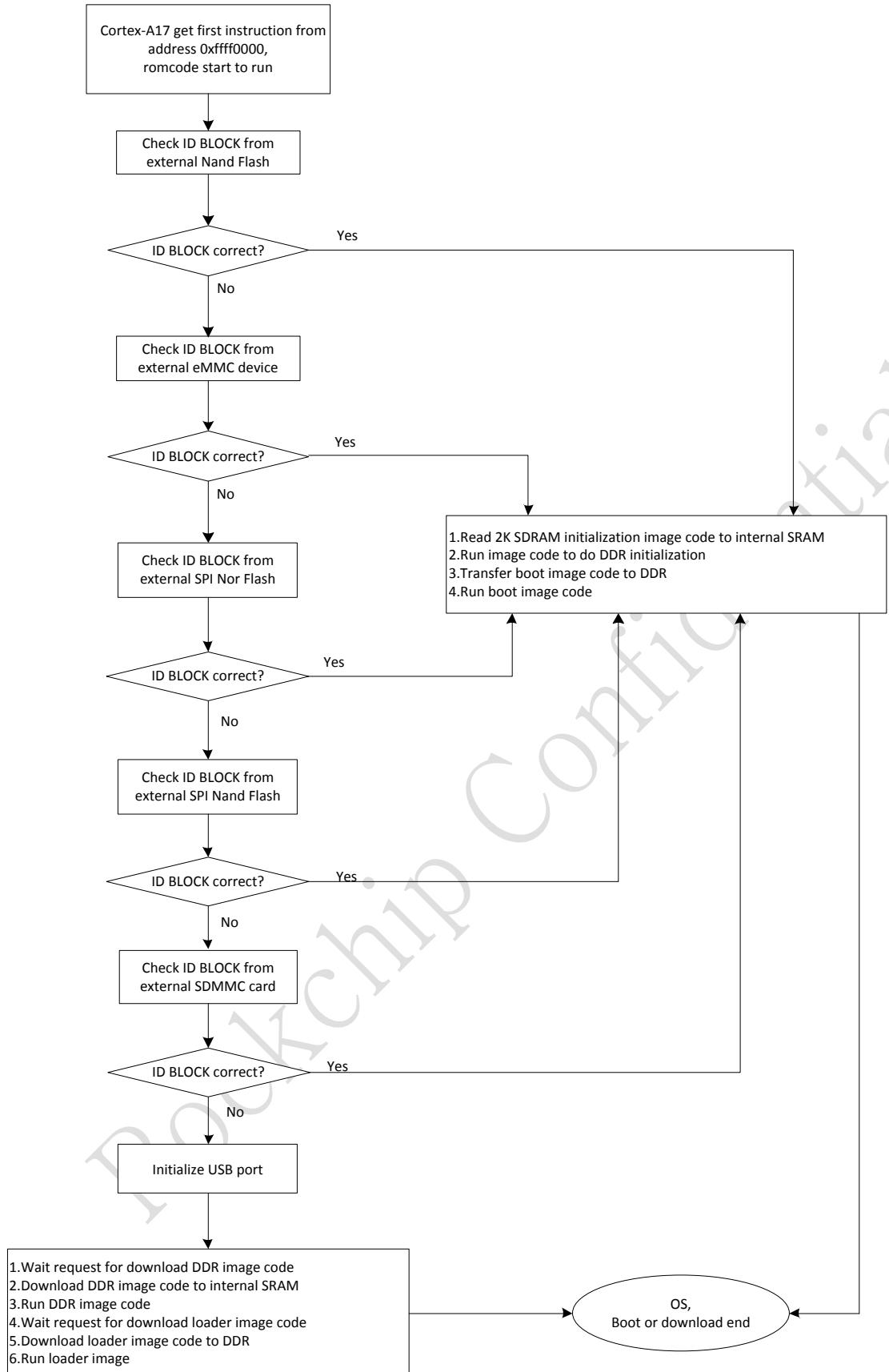


Fig. 2-2 RK3288 boot procedure flow

## 2.3 System Interrupt connection

RK3288 provides an general interrupt controller(GIC) for Cortex-A17 MPCore processor, which

has 112 SPI (shared peripheral interrupts) interrupt sources and 3 PPI(Private peripheral interrupt) interrupt source and separately generates one nIRQ and one nFIQ to CPU. The triggered type for each interrupts is high level sensitive, not programmable. The detailed interrupt sources connection is in the following table. For detailed GIC setting, please refer to Chapter 12.

Table 2-1 RK3288 Interrupt connection list

IRQ Type	IRQ ID	Source(spi)	Polarity
PPI	26	HYPERVERISOR TIMER	High level
	27	VIRTUAL TIMER	High level
	29	SECURE PHYSICAL TIMER	High level
	30	NON-SECURE PHY TIMER	High level
SPI	32	DMAC_BUS (0)	High level
	33	DMAC_BUS (1)	High level
	34	DMAC_PERI (0)	High level
	35	DMAC_PERI (1)	High level
	36	UPCTL 0	High level
	37	UPCTL 1	High level
	38	GPU_IRQJOB	High level
	39	GPU_IRQMMU	High level
	40	GPU_IRQGPU	High level
	41	VIDEO ENCODER	High level
	42	VIDEO DECODER	High level
	43	VIDEO MMU	High level
	44	HEVC	High level
	45	VIP	High level
	46	ISP	High level
	47	VOP_BIG	High level
	48	VOP_LIT	High level
	49	IEP	High level
	50	RGA	High level
	51	DSI 0 HOST	High level
	52	DSI 1 HOST	High level
	53	CSI HOST 0	High level
	54	CSI HOST 1	High level
	55	USB OTG	High level
	56	USB HOST 0 EHCI	High level
	57	USB HOST 1	High level
	58	HSIC	High level
	59	GMAC	High level
	60	GMAC PMT	High level
	61	GPS	High level
	62	GPS TIMER	High level
	63	HS-ADC/TSI	High level
	64	SD/MMC	High level
	65	SDIO 0	High level
	66	SDIO 1	High level

67	eMMC	High level
68	SARADC	High level
69	TSADC	High level
70	NANDC 0	High level
71	PERI MMU	High level
72	NANDC 1	High level
73	USB HOST 0 OHCI	High level
74	TPS	High level
75	SCR	High level
76	SPI0	High level
77	SPI1	High level
78	SPI2	High level
79	PS2C	High level
80	CRYPTO	High level
81	HOST PULSE 0	High level
82	HOST PULSE 1	High level
83	HOST 0	High level
84	HOST 1	High level
85	I2S/PCM (8ch)	High level
86	SPDIF(8ch)	High level
87	UART_BT	High level
88	UART_BB	High level
89	UART_DBG	High level
90	UART_GPS	High level
91	UART_EXP	High level
92	I2C_PMU	High level
93	I2C_AUDIO	High level
94	I2C_SENSOR	High level
95	I2C_CAM	High level
96	I2C_TP	High level
97	I2C_HDMI	High level
98	TIMER 6CH 0	High level
99	TIMER 6CH 1	High level
100	TIMER 6CH 2	High level
101	TIMER 6CH 3	High level
102	TIMER 6CH 4	High level
103	TIMER 6CH 5	High level
104	TIMER 2CH 0	High level
105	TIMER 2CH 1	High level
106	PWM0	High level
107	PWM1	High level
108	PWM2	High level
109	PWM3	High level
110	RK_PWM	High level
111	WDT	High level

	112	PMU	High level
	113	GPIO0	High level
	114	GPIO1	High level
	115	GPIO2	High level
	116	GPIO3	High level
	117	GPIO4	High level
	118	GPIO5	High level
	119	GPIO6	High level
	120	GPIO7	High level
	121	GPIO8	High level
	122	AHB ARBITER0 (USB)	High level
	123	AHB ARBITER1 (EMEM)	High level
	124	AHB ARBITER2 (MMC)	High level
	125	USBOTG_ID	High level
	126	USBOTG_BVALID	High level
	127	USBOTG_LINESTATE	High level
	128	USBHOST0_LINESTATE	High level
	129	USBHOST1_LINESTATE	High level
	130	eDP DP	High level
	131	SDMMC_DETECT_N	High level
	132	SDIO0_DETECT_N	High level
	133	SDIO1_DETECT_N	High level
	134	HDMI WAKEUP	High level
	135	HDMI	High level
	136	CCP	High level
	137	CCS	High level
	138	SDMMC DETECT DUAL EDGE	High level
	139	GPIO7_B3_DUAL_EDGE	High level
	140	GPIO7_C6_DUAL_EDGE	High level
	141	GPIO8_A2_DUAL_EDGE	High level
	142	eDP HDMI	High level
	143	HEVC MMU	High level
	186	PMUIRQ[0]	High level
	187	PMUIRQ[1]	High level
	188	PMUIRQ[2]	High level
	189	PMUIRQ[3]	High level

## 2.4 System DMA hardware request connection

RK3288 provides 2 DMA controllers: DMAC\_BUS inside bus system and DMAC\_PERI inside peripheral system. As for DMAC\_BUS, there are 6 hardware request ports. Another, 15 hardware request ports are used in DMAC\_PERI, the trigger type for each of them is high level, not programmable. For detailed descriptions of DMAC\_BUS and DMAC\_PERI, please refer to Chapter 10 and Chapter 11.

Table 2-2 RK3288 DMAC\_BUS Hardware request connection list

Req Number	Source	Polarity
0	I2S/PCM(8CH) TX	High level
1	I2S/PCM(8CH) RX	High level
2	SPDIF(2CH) TX	High level
3	SPDIF(8CH) TX	High level
4	UART_DBG TX	High level
5	UART_DBG RX	High level

Table 2-3 RK3288 DMAC\_PERI Hardware request connection list

Req Number	Source	Polarity
0	HS-ADC/TSI	High level
1	UART_BT TX	High level
2	UART_BT RX	High level
3	UART_BB TX	High level
4	UART_BB RX	High level
5	N/A	N/A
6	N/A	N/A
7	UART_GPS TX	High level
8	UART_GPS RX	High level
9	UART_EXP TX	High level
10	UART_EXP RX	High level
11	SPI0 TX	High level
12	SPI0 RX	High level
13	SPI1 TX	High level
14	SPI1 RX	High level
15	SPI2 TX	High level
16	SPI2 RX	High level

## Chapter 3 Clock & Reset Unit (CRU)

### 3.1 Overview

The CRU is an APB slave module that is designed for generating all of the system clocks, resets of chip. CRU generates system clock from PLL output clock or external clock source, and generates system reset from external power-on-reset, watchdog timer reset or software reset.

CRU supports the following features:

- Compliance to the AMBA APB interface
- Embedded five PLLs
- Flexible selection of clock source
- Supports the respective gating of all clocks
- Supports the respective software reset of all modules

### 3.2 Block Diagram

The CRU comprises with:

- PLL
- Register configuration unit
- Clock generate unit
- Reset generate unit

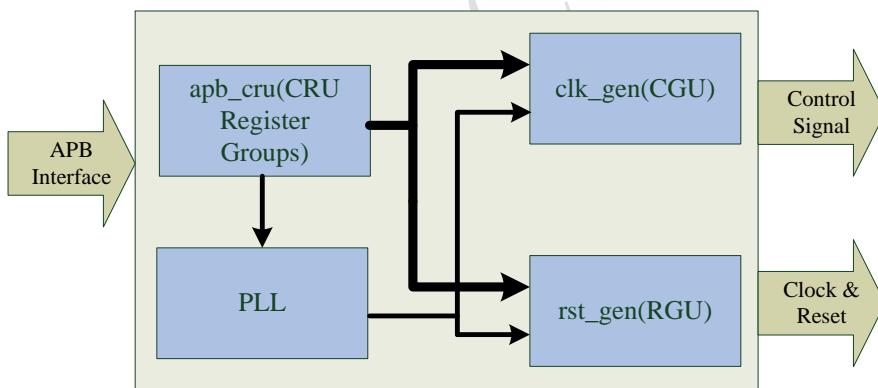


Fig. 3-1 CRU Architecture

### 3.3 System Clock Solution

#### 3.3.1 CRU architecture

The following diagrams show CRU clock architecture (mux and divider information).

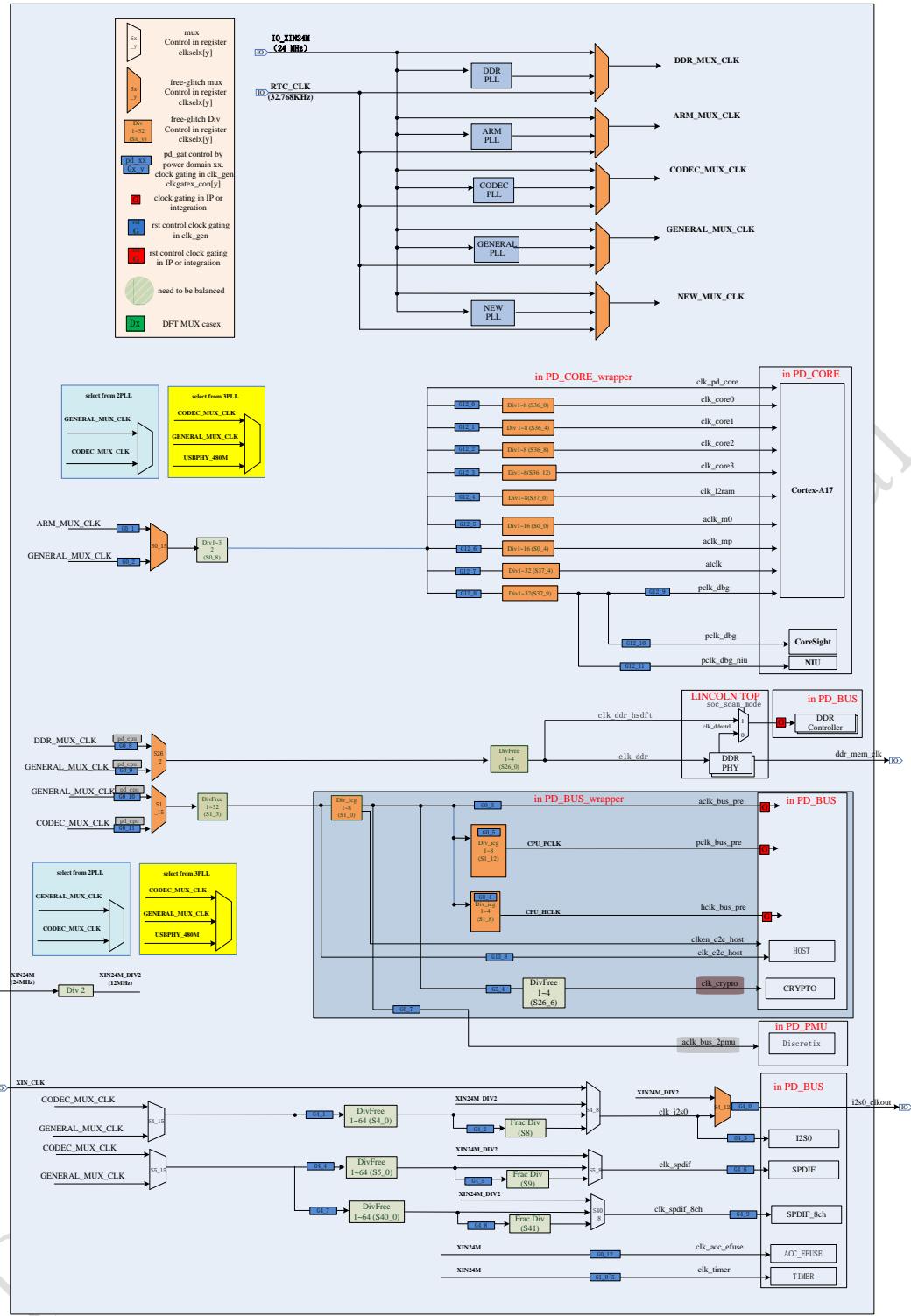


Fig. 3-2 CRU Clock Architecture Diagram 1

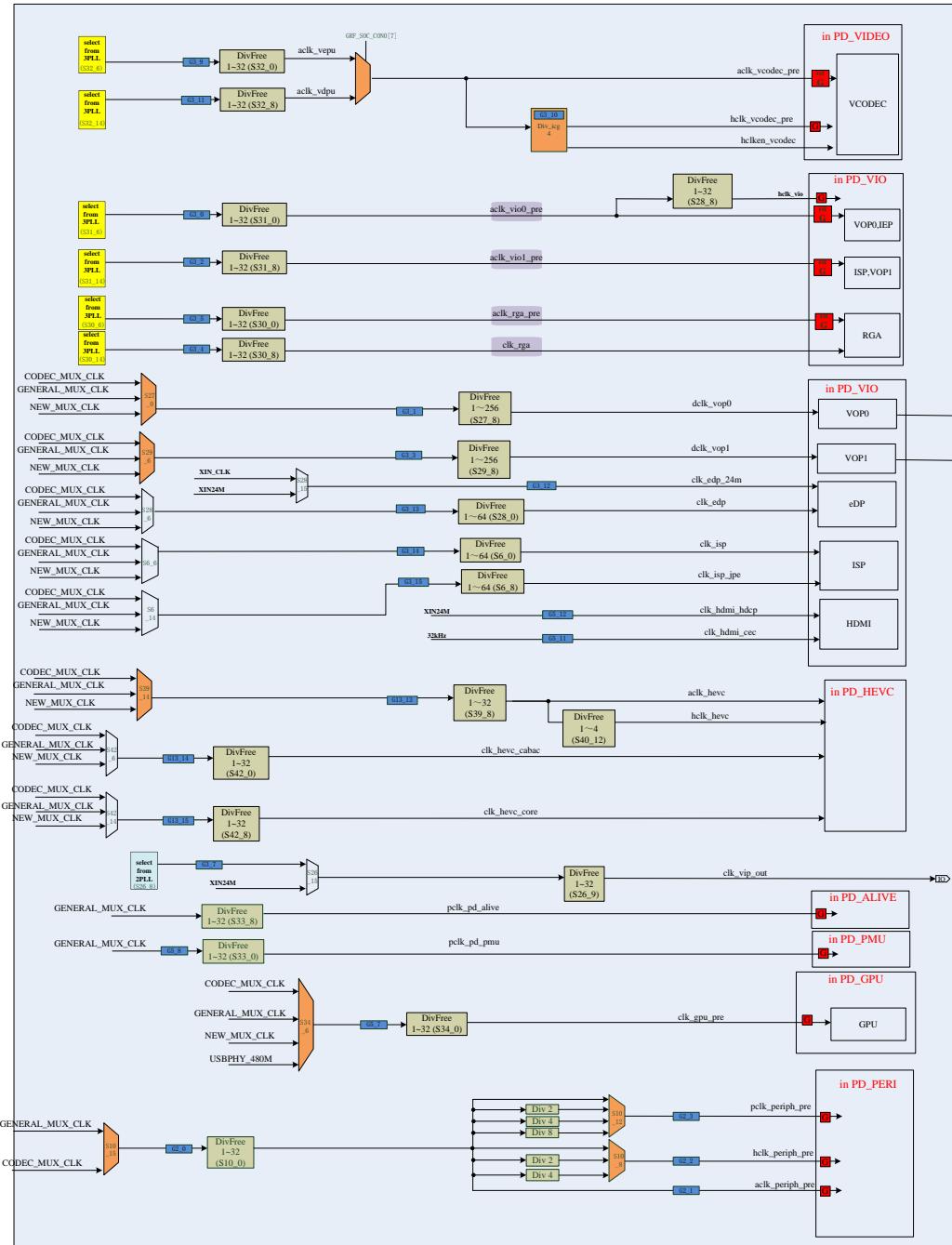


Fig. 3-3 CRU Clock Architecture Diagram 2

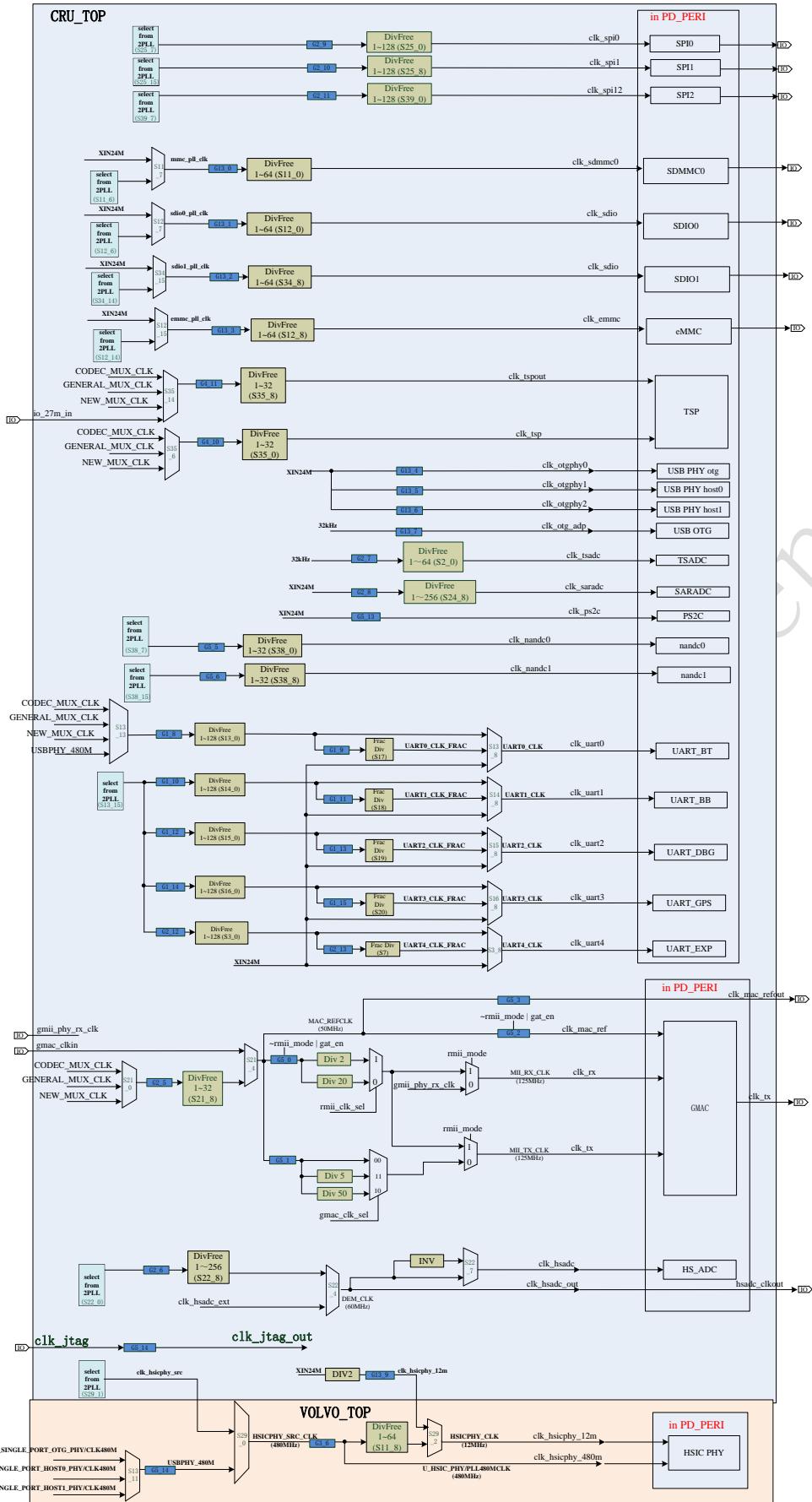


Fig. 3-4 CRU Clock Architecture Diagram 3

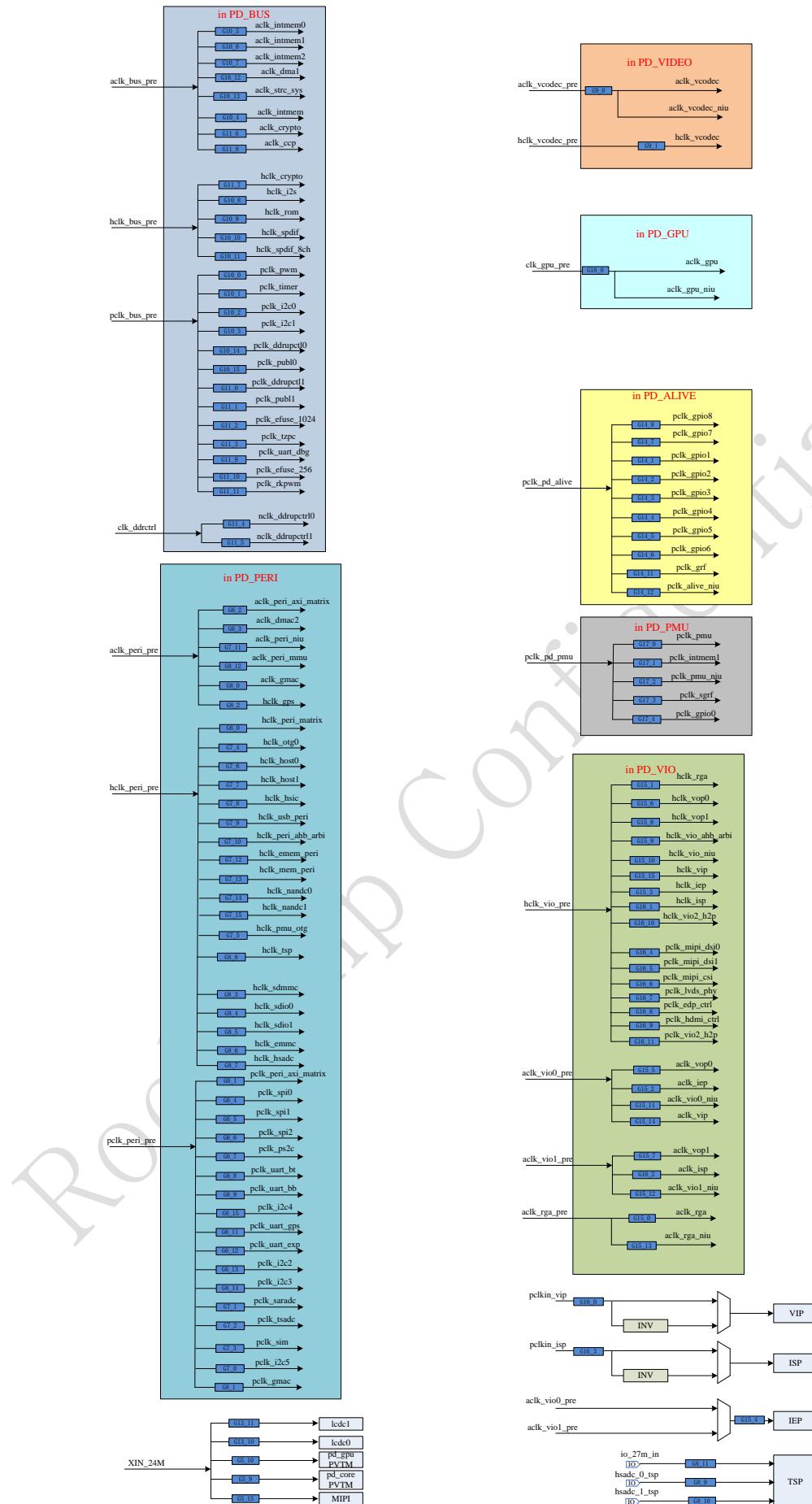


Fig. 3-5 CRU Clock Architecture Diagram 4

### 3.4 System Reset Solution

The following diagrams show reset architecture in this block.

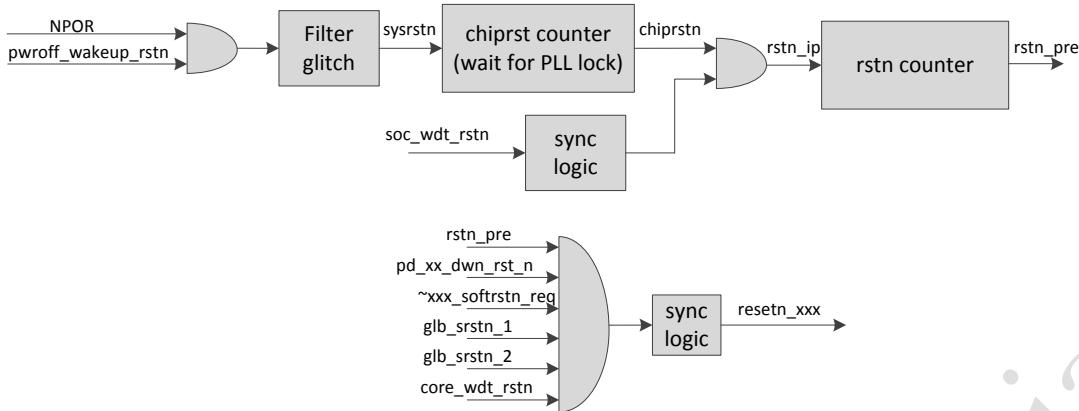


Fig. 3-6 Reset Architecture Diagram

Reset source of each reset signal includes hardware reset (NPOR), power-off mode wakeup reset (pwroff\_wakeup\_rstn), soc watch dog reset (soc\_wdt\_rstn), power domain power down reset (pd\_xx\_dwn\_RST\_N), software reset request (xxx\_softrstn\_req), global software reset1 (glb\_srstn\_1), global software reset2 (glb\_srstn\_2) and A9 core watch dog reset (core\_wdt\_rstn).

The 'xx' of pd\_xx\_dwn\_RST\_N represents core0, core1, core2, core3, cs, cpu, peri, vio, video or gpu. The 'xxx' of resetn\_xxx and xxx\_softrstn\_req is the module name.

Pwroff\_wakeup\_rstn is the reset when wakeup from the power-off mode, it will reset the all SOC logic except internal PMU.

Soc\_wdt\_rstn is the reset from watch-dog IP in the SoC, but core\_wdt\_rstn is the reset from A9 core watch-dog block.

Glb\_srstn\_1 and glb\_srstn\_2 are the global software reset by programming CRU register. When writing register CRU\_GLB\_SRST\_FST\_VALUE as 0xfd9, glb\_srstn\_1 will be asserted, and when writing register CRU\_GLB\_SRST\_SND\_VALUE as 0xea8, glb\_srstn\_2 will be asserted. The two software resets will be self-clear by hardware. Glb\_srstn\_1 will reset the all logic except PMU\_SYS\_REG0~3. And Glb\_srstn\_2 will reset the all logic except PMU\_SYS\_REG0~3, GRF and all GPIOs.

### 3.5 Function Description

There are five PLLs:

ARM PLL, DDR PLL, CODEC PLL, GENERAL PLL and NEW PLL in CRU.

PLLs all can be set to slow mode or deep slow mode, directly output selectable 24MHz or 32.768kHz. When power on or changing PLL setting, we must force PLL into slow mode to ensure output stable clock.

To maximize the flexibility, some of clocks can select divider source from three PLLs (CODEC PLL, GENERAL PLL and NEW PLL).

To provide some specific frequency, another solution is integrated: fractional divider. In order to be sure the performance for divided clock, there is some usage limit, we can only get low frequency and divider factor must be larger than 20.

All clocks can be software gated and all reset can be software generated.

## 3.6 PLL Introduction

### 3.6.1 Overview

This chip uses 2.2GHz PLL for all four PLLs. The 2.2GHz PLL is a general purpose, high-performance PLL-based clock generator. The VCO operates from 440 MHz to 2200MHz. It has a programmable output frequency, which ranges from 27.5 MHz to 2200 MHz configured through a 6-bit input divider, a 13-bit feedback divider and a 4-bit output divider. Around 50% duty cycle of output clocks can be achieved by enabling the output divider. It can also be used as a clock buffer through a bypass mode that bypasses and powers down the PLL. A full power-down mode is also available.

2.2GHz PLL supports the following features:

- Fully integrated, including loop filter
- Power supply: 1.0V single power supply
- VCO operating range: 440MHz – 2200MHz
- Output frequency range: 27.5MHz – 2200MHz
- Input frequency range: 269kHz – 2200MHz
- PFD comparison frequency range: 269kHz – 2200MHz
- Low power consumption: 3mA @ 1100MHz during normal operation
- Contains 6-bit input, 13-bit feedback and 4-bit output dividers
- Input divider value range: 1–64
- Feedback divider value range: 1–4096
- Output divider value range: 1, 2-16 (even only)
- Bandwidth adjustment of div. reference: 1–4096
- Output duty cycle: +/-5% (/1), +/-2% (/N)
- Period jitter (P-P) (max): +/-2.5% output cycle
- Reset pulse width (min): 5us
- Lock time (min allowed): 500 div. reference cycles
- Freq. overshoot (full~/half~/) (max): 40%/50%
- Ref. input jitter (long-term, P-P) (max): 2% div. reference cycle
- Reference H/L pulse width (min) : 230ps
- Bypass and Power-down mode
- Lock detector

### 3.6.2 Block diagram

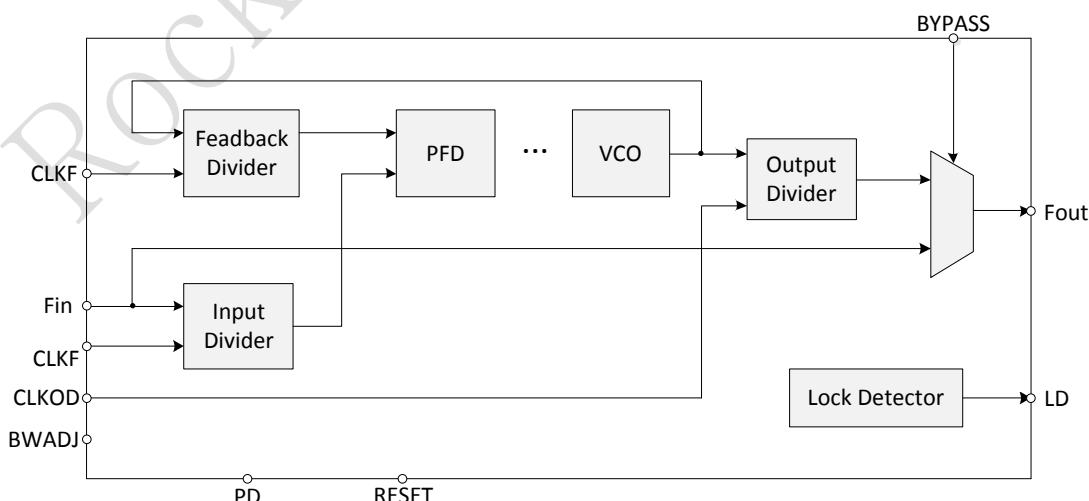


Fig. 3-7 PLL Block Diagram

### 3.6.3 Operation mode

#### A. Locked

The positive edges of the PLL feedback and reference signals are phase aligned in normal operation. Because the feedback signal is internal, NO phase relationship is guaranteed between RCLK and CLKOUT. The output clock frequency is programmable through the divider setting of CLKR[5:0], CLKF[12:0] and CLKOD[3:0].

#### B. Reset (RESET=1)

The PLL outputs a fixed free-running frequency in the range of 20MHz to 200MHz for a divide by 1 output depending on the specific PLL type.

#### C. Power-down (PWRDN=1)

All analog circuitry in the PLL is turned off so as to only dissipate leakage current. The digital dividers are not affected.

#### D. Bypass (BYPASS=1)

The reference input is bypassed directly to the outputs.

#### E. Test (TEST=1)

The reference input drives all dividers cascaded one after the other for production testing.

### 3.6.4 PLL Bandwidth Adjustment

The loop bandwidth (BW) of the PLL can be adjusted using BWADJ[11:0]. The bandwidth is given by:  $BW = nom\_BW * \sqrt{NF / 2 / NB}$ , where  $nom\_BW$  is approximately given by:  $nom\_BW = Fref / (NR * 20)$ , and  $Fref$  is the reference clock frequency. The damping factor (D) is approximately given by:  $D = nom\_D * \sqrt{NF / 2 / NB}$ , where  $nom\_D$  is approximately 1. Because the damping factor changes with bandwidth settings, the bandwidth is practically limited to:  $nom\_BW / \sqrt{2} < BW < nom\_BW * \sqrt{2}$ , in order to limit the damping factor range to 0.7 - 1.4. The -3dB bandwidth (Fbw\_3dB) is approximately given by:  $Fbw\_3dB = 2.4 * nom\_BW * (NF / 2 / NB)$ . The recommended setting for NB is NF / 2, which will yield the nominal bandwidth. Note that  $nom\_BW$  and  $nom\_D$  are chosen to result in optimal PLL loop dynamics.

## 3.7 Register Description

This section describes the control/status registers of the design.

### 3.7.1 CRU Registers Summary

Name	Offset	Size	Reset Value	Description
CRU_APOLL_CON0	0x0000	W	0x00000b01	ARM PLL configuration register0
CRU_APOLL_CON1	0x0004	W	0x000003e7	ARM PLL configuration register1
CRU_APOLL_CON2	0x0008	W	0x000001f3	ARM PLL configuration register2

Name	Offset	Size	Reset Value	Description
CRU_APPL_CON3	0x000c	W	0x00000008	ARM PLL configuration register3
CRU_DPLL_CON0	0x0010	W	0x00000b03	DDR PLL configuration register0
CRU_DPLL_CON1	0x0014	W	0x0000031f	DDR PLL configuration register1
CRU_DPLL_CON2	0x0018	W	0x0000018f	DDR PLL configuration register2
CRU_DPLL_CON3	0x001c	W	0x00000008	DDR PLL configuration register3
CRU_CPLL_CON0	0x0020	W	0x00000b03	CODEC PLL configuration register0
CRU_CPLL_CON1	0x0024	W	0x000002ff	CODEC PLL configuration register1
CRU_CPLL_CON2	0x0028	W	0x0000017f	CODEC PLL configuration register2
CRU_CPLL_CON3	0x002c	W	0x00000008	CODEC PLL configuration register3
CRU_GPLL_CON0	0x0030	W	0x00000b01	GENERAL PLL configuration register0
CRU_GPLL_CON1	0x0034	W	0x00000251	GENERAL PLL configuration register1
CRU_GPLL_CON2	0x0038	W	0x00000128	GENERAL PLL configuration register2
CRU_GPLL_CON3	0x003c	W	0x00000008	GENERAL PLL configuration register3
CRU_NPLL_CON0	0x0040	W	0x00000b03	NEW PLL configuration register0
CRU_NPLL_CON1	0x0044	W	0x000003e7	NEW PLL configuration register1
CRU_NPLL_CON2	0x0048	W	0x000001f3	NEW PLL configuration register2
CRU_NPLL_CON3	0x004c	W	0x00000008	NEW PLL configuration register3
CRU_MODE_CON	0x0050	W	0x00000000	System work mode control register
CRU_CLKSEL0_CON	0x0060	W	0x00000031	Internal clock select and divide register0
CRU_CLKSEL1_CON	0x0064	W	0x0000b109	Internal clock select and divide register1
CRU_CLKSEL2_CON	0x0068	W	0x00000020	Internal clock select and divide register2
CRU_CLKSEL3_CON	0x006c	W	0x00000200	Internal clock select and divide register3

Name	Offset	Size	Reset Value	Description
CRU_CLKSEL4_CON	0x0070	W	0x00000300	Internal clock select and divide register4
CRU_CLKSEL5_CON	0x0074	W	0x00000200	Internal clock select and divide register5
CRU_CLKSEL6_CON	0x0078	W	0x00000101	Internal clock select and divide register6
CRU_CLKSEL7_CON	0x007c	W	0x0bb8ea60	Internal clock select and divide register7
CRU_CLKSEL8_CON	0x0080	W	0x0bb8ea60	Internal clock select and divide register8
CRU_CLKSEL9_CON	0x0084	W	0x0bb8ea60	Internal clock select and divide register9
CRU_CLKSEL10_CON	0x0088	W	0x0000a101	Internal clock select and divide register10
CRU_CLKSEL11_CON	0x008c	W	0x00002780	Internal clock select and divide register11
CRU_CLKSEL12_CON	0x0090	W	0x00008080	Internal clock select and divide register12
CRU_CLKSEL13_CON	0x0094	W	0x00000200	Internal clock select and divide register13
CRU_CLKSEL14_CON	0x0098	W	0x00000200	Internal clock select and divide register14
CRU_CLKSEL15_CON	0x009c	W	0x00000200	Internal clock select and divide register15
CRU_CLKSEL16_CON	0x00a0	W	0x00000200	Internal clock select and divide register16
CRU_CLKSEL17_CON	0x00a4	W	0x0bb8ea60	Internal clock select and divide register17
CRU_CLKSEL18_CON	0x00a8	W	0x0bb8ea60	Internal clock select and divide register18
CRU_CLKSEL19_CON	0x00ac	W	0x0bb8ea60	Internal clock select and divide register19
CRU_CLKSEL20_CON	0x00b0	W	0x0bb8ea60	Internal clock select and divide register20
CRU_CLKSEL21_CON	0x00b4	W	0x00000b00	Internal clock select and divide register21
CRU_CLKSEL22_CON	0x00b8	W	0x00000900	Internal clock select and divide register22
CRU_CLKSEL23_CON	0x00bc	W	0x001f05dc	Internal clock select and divide register23
CRU_CLKSEL24_CON	0x00c0	W	0x00001700	Internal clock select and divide register24
CRU_CLKSEL25_CON	0x00c4	W	0x00000707	Internal clock select and divide register25

Name	Offset	Size	Reset Value	Description
CRU_CLKSEL26_CON	0x00c8	W	0x00000ec0	Internal clock select and divide register26
CRU_CLKSEL27_CON	0x00cc	W	0x00000700	Internal clock select and divide register27
CRU_CLKSEL28_CON	0x00d0	W	0x00000f03	Internal clock select and divide register28
CRU_CLKSEL29_CON	0x00d4	W	0x00000742	Internal clock select and divide register29
CRU_CLKSEL30_CON	0x00d8	W	0x00000000	Internal clock select and divide register30
CRU_CLKSEL31_CON	0x00dc	W	0x00000000	Internal clock select and divide register31
CRU_CLKSEL32_CON	0x00e0	W	0x00000101	Internal clock select and divide register32
CRU_CLKSEL33_CON	0x00e4	W	0x00000303	Internal clock select and divide register33
CRU_CLKSEL34_CON	0x00e8	W	0x00008000	Internal clock select and divide register34
CRU_CLKSEL35_CON	0x00ec	W	0x00000303	Internal clock select and divide register35
CRU_CLKSEL36_CON	0x00f0	W	0x00000000	Internal clock select and divide register36
CRU_CLKSEL37_CON	0x00f4	W	0x00001ef3	Internal clock select and divide register37
CRU_CLKSEL38_CON	0x00f8	W	0x00000303	Internal clock select and divide register38
CRU_CLKSEL39_CON	0x00fc	W	0x00000007	Internal clock select and divide register39
CRU_CLKSEL40_CON	0x0100	W	0x00000200	Internal clock select and divide register40
CRU_CLKSEL41_CON	0x0104	W	0x0bb8ea60	Internal clock select and divide register41
CRU_CLKSEL42_CON	0x0108	W	0x00000000	Internal clock select and divide register42
CRU_CLKGATE0_CO_N	0x0160	W	0x00000000	Internal clock gating control register0
CRU_CLKGATE1_CO_N	0x0164	W	0x00000000	Internal clock gating control register1
CRU_CLKGATE2_CO_N	0x0168	W	0x00000000	Internal clock gating control register2
CRU_CLKGATE3_CO_N	0x016c	W	0x00000000	Internal clock gating control register3
CRU_CLKGATE4_CO_N	0x0170	W	0x00000000	Internal clock gating control register4

Name	Offset	Size	Reset Value	Description
CRU_CLKGATE5_CO_N	0x0174	W	0x00000000	Internal clock gating control register5
CRU_CLKGATE6_CO_N	0x0178	W	0x00000000	Internal clock gating control register6
CRU_CLKGATE7_CO_N	0x017c	W	0x00000000	Internal clock gating control register7
CRU_CLKGATE8_CO_N	0x0180	W	0x00000000	Internal clock gating control register8
CRU_CLKGATE9_CO_N	0x0184	W	0x00000000	Internal clock gating control register9
CRU_CLKGATE10_CO_N	0x0188	W	0x00000000	Internal clock gating control register10
CRU_CLKGATE11_CO_N	0x018c	W	0x00000000	Internal clock gating control register11
CRU_CLKGATE12_CO_N	0x0190	W	0x00000000	Internal clock gating control register12
CRU_CLKGATE13_CO_N	0x0194	W	0x00000000	Internal clock gating control register13
CRU_CLKGATE14_CO_N	0x0198	W	0x00000000	Internal clock gating control register14
CRU_CLKGATE15_CO_N	0x019c	W	0x00000000	Internal clock gating control register15
CRU_CLKGATE16_CO_N	0x01a0	W	0x00000000	Internal clock gating control register16
CRU_CLKGATE17_CO_N	0x01a4	W	0x00000000	Internal clock gating control register17
CRU_CLKGATE18_CO_N	0x01a8	W	0x00000000	Internal clock gating control register18
CRU_GLB_SRST_FST_VALUE	0x01b0	W	0x00000000	The first global software reset config value
CRU_GLB_SRST_SN_D_VALUE	0x01b4	W	0x00000000	The second global software reset config value
CRU_SOFRST0_CO_N	0x01b8	W	0x00000000	Internal software reset control register0
CRU_SOFRST1_CO_N	0x01bc	W	0x00000000	Internal software reset control register1
CRU_SOFRST2_CO_N	0x01c0	W	0x00000000	Internal software reset control register2
CRU_SOFRST3_CO_N	0x01c4	W	0x00000000	Internal software reset control register3
CRU_SOFRST4_CO_N	0x01c8	W	0x00000000	Internal software reset control register4
CRU_SOFRST5_CO_N	0x01cc	W	0x00000000	Internal software reset control register5

Name	Offset	Size	Reset Value	Description
CRU_SOFRST6_CO_N	0x01d0	W	0x00000000	Internal software reset control register6
CRU_SOFRST7_CO_N	0x01d4	W	0x00000000	Internal software reset control register7
CRU_SOFRST8_CO_N	0x01d8	W	0x00000000	Internal software reset control register8
CRU_SOFRST9_CO_N	0x01dc	W	0x00000000	Internal software reset control register9
CRU_SOFRST10_CO_N	0x01e0	W	0x00000000	Internal software reset control register10
CRU_SOFRST11_CO_N	0x01e4	W	0x00000000	Internal software reset control register11
CRU_MISC_CON	0x01e8	W	0x00000000	SCU control register
CRU_GLB_CNT_TH	0x01ec	W	0x00000064	global reset wait counter threshold
CRU_GLB_RST_CON	0x01f0	W	0x00000000	global reset trigger select
CRU_GLB_RST_ST	0x01f8	W	0x00000000	global reset status
CRU_SDMMC_CON0	0x0200	W	0x00000002	sdmmc control0
CRU_SDMMC_CON1	0x0204	W	0x00000000	sdmmc control1
CRU_SDIO0_CON0	0x0208	W	0x00000002	sdio0 control0
CRU_SDIO0_CON1	0x020c	W	0x00000000	sdio0 control1
CRU_SDIO1_CON0	0x0210	W	0x00000002	sdio1 control0
CRU_SDIO1_CON1	0x0214	W	0x00000000	sdio1 control1
CRU_EMMC_CON0	0x0218	W	0x00000002	emmc control0
CRU_EMMC_CON1	0x021c	W	0x00000000	emmc control1

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

### 3.7.2 Detail Register Description

#### CRU\_APPL\_CON0

Address: Operational Base + offset (0x0000)

ARM PLL configuration register0

Bit	Attr	Reset Value	Description
31:30	RO	0x0	reserved
29:24	WO	0x00	clkr_mask CLKR value write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
23:20	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19:16	WO	0x0	clkod_mask Clock OD value write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved
13:8	RW	0x0b	clkr PLL CLKR factor control NR = CLKF + 1 NR: 1-64
7:4	RO	0x0	reserved
3:0	RW	0x1	clkod PLL CLKOD factor control NO = CLKOD + 1 NO: 1, 2-16 (even only)

**CRU\_APOLL\_CON1**

Address: Operational Base + offset (0x0004)

ARM PLL configuration register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	lock PLL lock status 1'b0: unlock 1'b1: lock
30:13	RO	0x0	reserved
12:0	RW	0x03e7	clkf PLL CLKF factor control NF = CLKF + 1 NF: 1-4096

**CRU\_APOLL\_CON2**

Address: Operational Base + offset (0x0008)

ARM PLL configuration register2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x1f3	bwadj PLL loop bandwidth adjust NB = BWADJ + 1

**CRU\_APOLL\_CON3**

Address: Operational Base + offset (0x000c)

ARM PLL configuration register3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
21	WO	0x0	reset_mask Reset configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
20	WO	0x0	test_mask Test configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
19	WO	0x0	ensat_mask Ensat configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
18	WO	0x0	fasten_mask Fasten configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
17	WO	0x0	power_down_mask Power down configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
16	WO	0x0	bypass_mask Bypass configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
15:6	RO	0x0	reserved
5	RW	0x0	reset PLL reset control 1'b0: normal 1'b1: reset
4	RW	0x0	test PLL test control 1'b0: normal 1'b1: test mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x1	ensat PLL saturation behavior enable 1'b0: disable 1'b1: enable
2	RW	0x0	fasten PLL enable fast locking circuit 1'b0: disable 1'b1: enable
1	RW	0x0	power_down PLL power down control 1'b0: no power down 1'b1: power down
0	RW	0x0	bypass PLL bypass mode control 1'b0: no bypass 1'b1: bypass

**CRU\_DPLL\_CON0**

Address: Operational Base + offset (0x0010)

DDR PLL configuration register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved
29:24	WO	0x00	clkr_mask CLKR value write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
23:20	RO	0x0	reserved
19:16	WO	0x0	clkod_mask Clock OD value write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved
13:8	RW	0x0b	clkr PLL CLKR factor control NR = CLKF + 1 NR: 1-64
7:4	RO	0x0	reserved
3:0	RW	0x3	clkod PLL CLKOD factor control NO = CLKOD + 1 NO: 1, 2-16 (even only)

**CRU\_DPLL\_CON1**

Address: Operational Base + offset (0x0014)

DDR PLL configuration register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	lock PLL lock status 1'b0: unlock 1'b1: lock
30:13	RO	0x0	reserved
12:0	RW	0x031f	clkf PLL CLKF factor control NF = CLKF + 1 NF: 1-4096

**CRU\_DPLL\_CON2**

Address: Operational Base + offset (0x0018)

DDR PLL configuration register2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x18f	bwadj PLL loop bandwidth adjust NB = BWADJ + 1

**CRU\_DPLL\_CON3**

Address: Operational Base + offset (0x001c)

DDR PLL configuration register3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RO	0x0	reserved
21	WO	0x0	reset_mask Reset configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
20	WO	0x0	test_mask Test configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19	WO	0x0	ensat_mask Ensat configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
18	WO	0x0	fasten_mask Fasten configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
17	WO	0x0	power_down_mask Power down configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
16	WO	0x0	bypass_mask Bypass configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
15:6	RO	0x0	reserved
5	RW	0x0	reset PLL reset control 1'b0: normal 1'b1: reset
4	RW	0x0	test PLL test control 1'b0: normal 1'b1: test mode
3	RW	0x1	ensat PLL saturation behavior enable 1'b0: disable 1'b1: enable
2	RW	0x0	fasten PLL enable fast locking circuit 1'b0: disable 1'b1: enable
1	RW	0x0	power_down PLL power down control 1'b0: no power down 1'b1: power down

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	bypass PLL bypass mode control 1'b0: no bypass 1'b1: bypass

**CRU\_CPLL\_CON0**

Address: Operational Base + offset (0x0020)

CODEC PLL configuration register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved
29:24	WO	0x00	clk_r_mask CLKR value write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
23:20	RO	0x0	reserved
19:16	WO	0x0	clkod_mask Clock OD value write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved
13:8	RW	0x0b	clk_r PLL CLKR factor control NR = CLKF + 1 NR: 1-64
7:4	RO	0x0	reserved
3:0	RW	0x3	clkod PLL CLKOD factor control NO = CLKOD + 1 NO: 1, 2-16 (even only)

**CRU\_CPLL\_CON1**

Address: Operational Base + offset (0x0024)

CODEC PLL configuration register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	lock PLL lock status 1'b0: unlock 1'b1: lock
30:13	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12:0	RW	0x02ff	clkf PLL CLKF factor control NF = CLKF + 1 NF: 1-4096

**CRU\_CPLL\_CON2**

Address: Operational Base + offset (0x0028)

CODEC PLL configuration register2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x17f	bwadj PLL loop bandwidth adjust NB = BWADJ + 1

**CRU\_CPLL\_CON3**

Address: Operational Base + offset (0x002c)

CODEC PLL configuration register3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RO	0x0	reserved
21	WO	0x0	reset_mask Reset configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
20	WO	0x0	test_mask Test configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
19	WO	0x0	ensat_mask Ensat configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
18	WO	0x0	fasten_mask Fasten configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17	WO	0x0	power_down_mask Power down configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
16	WO	0x0	bypass_mask Bypass configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
15:6	RO	0x0	reserved
5	RW	0x0	reset PLL reset control 1'b0: normal 1'b1: reset
4	RW	0x0	test PLL test control 1'b0: normal 1'b1: test mode
3	RW	0x1	ensat PLL saturation behavior enable 1'b0: disable 1'b1: enable
2	RW	0x0	fasten PLL enable fast locking circuit 1'b0: disable 1'b1: enable
1	RW	0x0	power_down PLL power down control 1'b0: no power down 1'b1: power down
0	RW	0x0	bypass PLL bypass mode control 1'b0: no bypass 1'b1: bypass

**CRU\_GPLL\_CON0**

Address: Operational Base + offset (0x0030)

GENERAL PLL configuration register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
29:24	WO	0x00	clk_r_mask CLKR value write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
23:20	RO	0x0	reserved
19:16	WO	0x0	clkod_mask Clock OD value write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved
13:8	RW	0x0b	clk_r PLL CLKR factor control NR = CLKF + 1 NR: 1-64
7:4	RO	0x0	reserved
3:0	RW	0x1	clkod PLL CLKOD factor control NO = CLKOD + 1 NO: 1, 2-16 (even only)

**CRU\_GPLL\_CON1**

Address: Operational Base + offset (0x0034)

GENERAL PLL configuration register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	lock PLL lock status 1'b0: unlock 1'b1: lock
30:13	RO	0x0	reserved
12:0	RW	0x0251	clkf PLL CLKF factor control NF = CLKF + 1 NF: 1-4096

**CRU\_GPLL\_CON2**

Address: Operational Base + offset (0x0038)

GENERAL PLL configuration register2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:0	RW	0x128	bwadj PLL loop bandwidth adjust NB = BWADJ + 1

**CRU\_GPLL\_CON3**

Address: Operational Base + offset (0x003c)

GENERAL PLL configuration register3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RO	0x0	reserved
21	WO	0x0	reset_mask Reset configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
20	WO	0x0	test_mask Test configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
19	WO	0x0	ensat_mask Ensat configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
18	WO	0x0	fasten_mask Fasten configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
17	WO	0x0	power_down_mask Power down configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
16	WO	0x0	bypass_mask Bypass configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:6	RO	0x0	reserved
5	RW	0x0	reset PLL reset control 1'b0: normal 1'b1: reset
4	RW	0x0	test PLL test control 1'b0: normal 1'b1: test mode
3	RW	0x1	ensat PLL saturation behavior enable 1'b0: disable 1'b1: enable
2	RW	0x0	fasten PLL enable fast locking circuit 1'b0: disable 1'b1: enable
1	RW	0x0	power_down PLL power down control 1'b0: no power down 1'b1: power down
0	RW	0x0	bypass PLL bypass mode control 1'b0: no bypass 1'b1: bypass

**CRU\_NPLL\_CON0**

Address: Operational Base + offset (0x0040)

NEW PLL configuration register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved
29:24	WO	0x00	clkr_mask CLKR value write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
23:20	RO	0x0	reserved
19:16	WO	0x0	clkod_mask Clock OD value write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13:8	RW	0x0b	clkf PLL CLKF factor control NR = CLKF + 1 NR: 1-64
7:4	RO	0x0	reserved
3:0	RW	0x3	clkod PLL CLKOD factor control NO = CLKOD + 1 NO: 1, 2-16 (even only)

**CRU\_NPLL\_CON1**

Address: Operational Base + offset (0x0044)

NEW PLL configuration register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	lock PLL lock status 1'b0: unlock 1'b1: lock
30:13	RO	0x0	reserved
12:0	RW	0x03e7	clkf PLL CLKF factor control NF = CLKF + 1 NF: 1-4096

**CRU\_NPLL\_CON2**

Address: Operational Base + offset (0x0048)

NEW PLL configuration register2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x1f3	bwadj PLL loop bandwidth adjust NB = BWADJ + 1

**CRU\_NPLL\_CON3**

Address: Operational Base + offset (0x004c)

NEW PLL configuration register3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RO	0x0	reserved
21	WO	0x0	reset_mask Reset configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
20	WO	0x0	test_mask Test configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
19	WO	0x0	ensat_mask Ensat configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
18	WO	0x0	fasten_mask Fasten configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
17	WO	0x0	power_down_mask Power down configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
16	WO	0x0	bypass_mask Bypass configuration write mask. When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit
15:6	RO	0x0	reserved
5	RW	0x0	reset PLL reset control 1'b0: normal 1'b1: reset
4	RW	0x0	test PLL test control 1'b0: normal 1'b1: test mode
3	RW	0x1	ensat PLL saturation behavior enable 1'b0: disable 1'b1: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x0	fasten PLL enable fast locking circuit 1'b0: disable 1'b1: enable
1	RW	0x0	power_down PLL power down control 1'b0: no power down 1'b1: power down
0	RW	0x0	bypass PLL bypass mode control 1'b0: no bypass 1'b1: bypass

**CRU\_MODE\_CON**

Address: Operational Base + offset (0x0050)

System work mode control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RW	0x0	npll_work_mode NEW PLL work mode select 2'b00: Slow mode, clock from external 24MHz OSC (default) 2'b01: Normal mode, clock from PLL output 2'b10: Deep slow mode, clock from external 32.768kHz
13:12	RW	0x0	gpll_work_mode GENERAL PLL work mode select 2'b00: Slow mode, clock from external 24MHz OSC (default) 2'b01: Normal mode, clock from PLL output 2'b10: Deep slow mode, clock from external 32.768kHz
11:10	RO	0x0	reserved
9:8	RW	0x0	cpll_work_mode CODEC PLL work mode select 2'b00: Slow mode, clock from external 24MHz OSC (default) 2'b01: Normal mode, clock from PLL output 2'b10: Deep slow mode, clock from external 32.768kHz

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:6	RO	0x0	reserved
5:4	RW	0x0	dpll_work_mode DDR PLL work mode select 2'b00: Slow mode, clock from external 24MHz OSC (default) 2'b01: Normal mode, clock from PLL output 2'b10: Deep slow mode, clock from external 32.768kHz
3:2	RO	0x0	reserved
1:0	RW	0x0	apll_work_mode ARM PLL work mode select 2'b00: Slow mode, clock from external 24MHz OSC (default) 2'b01: Normal mode, clock from PLL output 2'b10: Deep slow mode, clock from external 32.768kHz

**CRU\_CLKSEL0\_CON**

Address: Operational Base + offset (0x0060)

Internal clock select and divide register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	core_clk_pll_sel CORE clock pll source selection 1'b0: select ARM PLL 1'b1: select GENERAL PLL
14:13	RO	0x0	reserved
12:8	RW	0x00	a17_core_div_con Control A17 core clock divider frequency $clk_{core} = clk_{src}/(div\_con+1)$
7:4	RW	0x3	aclk_core_mp_div_con Control core MP AXI clock divider frequency $clk = clk_{src}/(div\_con+1)$
3:0	RW	0x1	aclk_core_m0_div_con Control core M0 AXI clock divider frequency $clk = clk_{src}/(div\_con+1)$

**CRU\_CLKSEL1\_CON**

Address: Operational Base + offset (0x0064)

Internal clock select and divide register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x1	bus_aclk_pll_sel pd_bus axi clock pll source selection 1'b0: select CODEC PLL 1'b1: select GENERAL PLL
14:12	RW	0x3	pd_bus_pclk_div_con Control pd_bus APB clock divider frequency $clk=clk\_src/(div\_con+1)$
11:10	RO	0x0	reserved
9:8	RW	0x1	pd_bus_hclk_div_con Control pd_bus AHB clock divider frequency 2'b00: aclk_bus:hclk_bus = 1:1 2'b01: aclk_bus:hclk_bus = 2:1 2'b11: aclk_bus:hclk_bus = 4:1
7:3	RW	0x01	pd_bus_aclk_div_con Control pd_bus aclk divider frequency $clk=clk\_src/(div\_con+1)$
2:0	RW	0x1	pd_bus_clk_div_con1 Control pd_bus AXI clock divider1 frequency $clk=clk\_src/(div\_con+1)$

**CRU\_CLKSEL2\_CON**

Address: Operational Base + offset (0x0068)

Internal clock select and divide register2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:13	RO	0x0	reserved
12:8	RW	0x00	testout_div_con test out clk divider frequency $clk\_testout=testout\_clk\_src/(testout\_div\_con+1)$
7:6	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:0	RW	0x20	tsadc_div_con Control tsadc divider frequency $\text{clk\_tsadc} = \text{tsadc\_clk\_src} / (\text{tsadc\_div\_con} + 1)$

**CRU\_CLKSEL3\_CON**

Address: Operational Base + offset (0x006c)

Internal clock select and divide register3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:10	RO	0x0	reserved
9:8	RW	0x2	uart4_clk_sel Control UART4 clock work frequency selection 2'b00: select divider output from pll divider 2'b01: select divider output from fraction divider 2'b10: select 24MHz from osc input
7	RO	0x0	reserved
6:0	RW	0x00	uart4_div_con Control UART4 divider frequency $\text{clk\_uart0} = \text{uart\_clk\_src} / (\text{uart0\_div\_con} + 1)$

**CRU\_CLKSEL4\_CON**

Address: Operational Base + offset (0x0070)

Internal clock select and divide register4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	i2s_pll_sel Control I2S PLL source selection 1'b0: select codec pll clock 1'b1: select general pll clock
14:13	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x0	i2s0_outclk_sel Control I2S clock work frequency selection 1'b0: select clk_i2s 1'b1: select 12MHz
11:10	RO	0x0	reserved
9:8	RW	0x3	i2s0_clk_sel Control I2S clock work frequency selection 2'b00: select divider output from pll divider 2'b01: select divider output from fraction divider 2'b10: select clock from IO input 2'b11: select 12MHz from osc input
7	RO	0x0	reserved
6:0	RW	0x00	i2s0_pll_div_con Control I2S PLL output divider frequency $i2s1\_div\_clk = i2s1\_div\_src / (i2s1\_pll\_div\_con + 1)$

**CRU\_CLKSEL5\_CON**

Address: Operational Base + offset (0x0074)

Internal clock select and divide register5

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	spdif_pll_sel Control SPDIF PLL source selection 1'b0: select codec pll clock 1'b1: select general pll clock
14:10	RO	0x0	reserved
9:8	RW	0x2	spdif_clk_sel Control SPDIF clock work frequency selection 2'b00: select divider output from pll divider 2'b01: select divider output from fraction divider 2'b10: select 12MHz from osc input
7	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6:0	RW	0x00	spdif_pll_div_con Control SPDIF PLL output divider frequency $\text{spdif\_div\_clk} = \text{spdif\_div\_src} / (\text{spdif\_pll\_div\_con} + 1)$

**CRU\_CLKSEL6\_CON**

Address: Operational Base + offset (0x0078)

Internal clock select and divide register6

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RW	0x0	isp_jpeg_pll_sel Control ISP jpeg PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select new pll clock
13:8	RW	0x01	isp_jpeg_div_con Control isp jpeg divider frequency $\text{jpeg\_div\_clk} = \text{jpeg\_div\_src} / (\text{isp\_jpeg\_div\_con} + 1)$
7:6	RW	0x0	isp_pll_sel Control ISP PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select new pll clock
5:0	RW	0x01	isp_div_con Control isp divider frequency $\text{isp\_div\_clk} = \text{isp\_div\_src} / (\text{isp\_pll\_div\_con} + 1)$

**CRU\_CLKSEL7\_CON**

Address: Operational Base + offset (0x007c)

Internal clock select and divide register7

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x0bb8ea60	uart4_frac_factor Control uart4 fraction divider frequency High 16-bit for numerator Low 16-bit for denominator

**CRU\_CLKSEL8\_CON**

Address: Operational Base + offset (0x0080)

Internal clock select and divide register8

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x0bb8ea60	i2s0_frac_factor Control I2S fraction divider frequency High 16-bit for numerator Low 16-bit for denominator

**CRU\_CLKSEL9\_CON**

Address: Operational Base + offset (0x0084)

Internal clock select and divide register9

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x0bb8ea60	spdif_frac_factor Control SPDIF fraction divider frequency High 16-bit for numerator Low 16-bit for denominator

**CRU\_CLKSEL10\_CON**

Address: Operational Base + offset (0x0088)

Internal clock select and divide register10

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x1	peri_pll_sel Control peripheral clock PLL source selection 1'b0: select codec pll clock 1'b1: select general pll clock
14	RO	0x0	reserved
13:12	RW	0x2	peri_pclk_div_con Control the divider ratio between aclk_periph and pclk_periph 2'b00: aclk_periph:pclk_periph = 1:1 2'b01: aclk_periph:pclk_periph = 2:1 2'b10: aclk_periph:pclk_periph = 4:1 2'b11: aclk_periph:pclk_periph = 8:1
11:10	RO	0x0	reserved
9:8	RW	0x1	peri_hclk_div_con Control the divider ratio between aclk_periph and hclk_periph 2'b00: aclk_periph:hclk_periph = 1:1 2'b01: aclk_periph:hclk_periph = 2:1 2'b10: aclk_periph:hclk_periph = 4:1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:5	RO	0x0	reserved
4:0	RW	0x01	peri_aclk_div_con Control peripheral clock divider frequency aclk_periph=periph_clk_src/(peri_aclk_div_c on+1)

**CRU\_CLKSEL11\_CON**

Address: Operational Base + offset (0x008c)

Internal clock select and divide register11

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved
13:8	RW	0x27	hsicphy_div_con Control HSICPHY divider frequency $clk_{hsicphy\_12m} = clk_{hsicphy\_480m}/(hsicphy\_div\_con+1)$
7:6	RW	0x2	mmc0_pll_sel Control mmc0 clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select 24MHz
5:0	RW	0x00	mmc0_div_con Control SDMMC0 divider frequency $clk_{sdmmc0} = general\_pll\_clk/(mmc0\_div\_con+1)$

**CRU\_CLKSEL12\_CON**

Address: Operational Base + offset (0x0090)

Internal clock select and divide register12

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:14	RW	0x2	emmc_pll_sel Control emmc clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select 24MHz
13:8	RW	0x00	emmc_div_con Control EMMC divider frequency $clk_{emmc} = general\_pll\_clk / (emmc\_div\_con + 1)$
7:6	RW	0x2	sdio0_pll_sel Control sdio0 clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select 24MHz
5:0	RW	0x00	sdio0_div_con Control SDIO0 divider frequency $clk_{sdio} = general\_pll\_clk / (sdio\_div\_con + 1)$

**CRU\_CLKSEL13\_CON**

Address: Operational Base + offset (0x0094)

Internal clock select and divide register13

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	uart_pll_sel Control UART1~4 clock PLL source selection 1'b0: select codec pll clock 1'b1: select general pll clock
14:13	RW	0x0	uart0_src_sel UART0 clock source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select 480M USBPHY clock 2'b11: select new pll clock
12:11	RW	0x0	usbphy_480m_sel USBPHY 480M clock source selection 2'b00: select HOST0 USB pll clock 2'b01: select HOST1 USB pll clock 2'b10: select OTG USB pll clock
10	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9:8	RW	0x2	uart0_clk_sel Control UART0 clock work frequency selection 2'b00: select divider output from pll divider 2'b01: select divider output from fraction divider 2'b10: select 24MHz from osc input
7	RO	0x0	reserved
6:0	RW	0x00	uart0_div_con Control UART0 divider frequency $\text{clk\_uart0} = \text{uart\_clk\_src} / (\text{uart0\_div\_con} + 1)$

**CRU\_CLKSEL14\_CON**

Address: Operational Base + offset (0x0098)

Internal clock select and divide register14

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:10	RO	0x0	reserved
9:8	RW	0x2	uart1_clk_sel Control UART1 clock work frequency selection 2'b00: select divider output from pll divider 2'b01: select divider output from fraction divider 2'b10: select 24MHz from osc input
7	RO	0x0	reserved
6:0	RW	0x00	uart1_div_con Control UART1 divider frequency $\text{clk\_uart1} = \text{uart\_clk\_src} / (\text{uart1\_div\_con} + 1)$

**CRU\_CLKSEL15\_CON**

Address: Operational Base + offset (0x009c)

Internal clock select and divide register15

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:10	RO	0x0	reserved
9:8	RW	0x2	uart2_clk_sel Control UART2 clock work frequency selection 2'b00: select divider output from pll divider 2'b01: select divider output from fraction divider 2'b10: select 24MHz from osc input
7	RO	0x0	reserved
6:0	RW	0x00	uart2_div_con Control UART2 divider frequency $\text{clk\_uart2} = \text{uart\_clk\_src} / (\text{uart2\_div\_con} + 1)$

**CRU\_CLKSEL16\_CON**

Address: Operational Base + offset (0x00a0)

Internal clock select and divide register16

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:10	RO	0x0	reserved
9:8	RW	0x2	uart3_clk_sel Control UART3 clock work frequency selection 2'b00: select divider output from pll divider 2'b01: select divider output from fraction divider 2'b10: select 24MHz from osc input
7	RO	0x0	reserved
6:0	RW	0x00	uart3_div_con Control UART3 divider frequency $\text{clk\_uart3} = \text{uart\_clk\_src} / (\text{uart3\_div\_con} + 1)$

**CRU\_CLKSEL17\_CON**

Address: Operational Base + offset (0x00a4)

Internal clock select and divide register17

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x0bb8ea60	uart0_frac_factor Control UART0 fraction divider frequency High 16-bit for numerator Low 16-bit for denominator

**CRU\_CLKSEL18\_CON**

Address: Operational Base + offset (0x00a8)

Internal clock select and divide register18

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x0bb8ea60	uart1_frac_factor Control UART1 fraction divider frequency High 16-bit for numerator Low 16-bit for denominator

**CRU\_CLKSEL19\_CON**

Address: Operational Base + offset (0x00ac)

Internal clock select and divide register19

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x0bb8ea60	uart2_frac_factor Control UART2 fraction divider frequency High 16-bit for numerator Low 16-bit for denominator

**CRU\_CLKSEL20\_CON**

Address: Operational Base + offset (0x00b0)

Internal clock select and divide register20

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x0bb8ea60	uart3_frac_factor Control UART3 fraction divider frequency High 16-bit for numerator Low 16-bit for denominator

**CRU\_CLKSEL21\_CON**

Address: Operational Base + offset (0x00b4)

Internal clock select and divide register21

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:13	RO	0x0	reserved
12:8	RW	0x0b	mac_div_con Control EMAC divider frequency $clk\_mac\_ref = mac\_clk\_src / (mac\_div\_con + 1)$
7:5	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RW	0x0	rmii_extclk_sel Control RMII external clock selection 1'b0: select internal divider clock 1'b1: select external input clock
3:2	RO	0x0	reserved
1:0	RW	0x0	mac_pll_sel Control EMAC clock PLL source selection 2'b00: select new pll clock 2'b01: select codec pll clock 2'b10: select general pll clock

**CRU\_CLKSEL22\_CON**

Address: Operational Base + offset (0x00b8)

Internal clock select and divide register22

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:8	RW	0x09	hsadc_div_con Control HSADC divider frequency $clk_{hsadc} = clk_{src}/(hsadc\_div\_con + 1)$
7	RW	0x0	hsadc_inv_sel Control HSADC inverter clock 1'b0: select buffer output 1'b1: select inverter output
6:5	RO	0x0	reserved
4	RW	0x0	hsadc_clk_sel Control HSADC clock work frequency selection 1'b0: select divider ouput from pll divider 1'b1: select external input clock
3:2	RO	0x0	reserved
1	RW	0x0	wifi_pll_sel Control wifi clock PLL source selection 1'b0: select codec pll clock 1'b1: select general pll clock
0	RW	0x0	hsadc_pll_sel Control HSADC clock PLL source selection 1'b0: select codec pll clock 1'b1: select general pll clock

**CRU\_CLKSEL23\_CON**

Address: Operational Base + offset (0x00bc)

Internal clock select and divide register23

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x001f05dc	wifi_frac_factor Control wifi fraction divider frequency High 16-bit for numerator Low 16-bit for denominator

**CRU\_CLKSEL24\_CON**

Address: Operational Base + offset (0x00c0)

Internal clock select and divide register24

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:8	RW	0x17	saradc_div_con Control SARADC clock divider frequency $clk_{saradc}=24MHz/(saradc\_div\_con+1)$
7:0	RO	0x0	reserved

**CRU\_CLKSEL25\_CON**

Address: Operational Base + offset (0x00c4)

Internal clock select and divide register25

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	spi1_pll_sel Control spi1 clock PLL source selection 1'b0: select codec pll clock 1'b1: select general pll clock
14:8	RW	0x07	spi1_div_con Control SPI1 clock divider frequency $clk_{spi1}=general\_pll\_clk/(spi1\_div\_con+1)$
7	RW	0x0	spi0_pll_sel Control spi0 clock PLL source selection 1'b0: select codec pll clock 1'b1: select general pll clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6:0	RW	0x07	spi0_div_con Control SPI0 clock divider frequency $\text{clk\_spi0} = \text{general\_pll\_clk}/(\text{spi0\_div\_con}+1)$

**CRU\_CLKSEL26\_CON**

Address: Operational Base + offset (0x00c8)

Internal clock select and divide register26

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	cif_clk_out_sel CIF clock output selection 1'b0: select PLL divout 1'b1: select 24MHz
14	RO	0x0	reserved
13:9	RW	0x07	cif_clk_div_con cif clock divider frequency $\text{clk} = \text{clk\_src}/(\text{div\_con}+1)$
8	RW	0x0	cif_clk_pll_sel CIF clock pll source selection 1'b0: select codec PLL 1'b1: select general PLL
7:6	RW	0x3	crypto_div_con crypto clock divider frequency $\text{clk} = \text{clk\_src}/(\text{div\_con}+1)$
5:3	RO	0x0	reserved
2	RW	0x0	ddr_clk_pll_sel DDR clock pll source selection 1'b0: select DDR PLL 1'b1: select GENERAL PLL
1:0	RW	0x0	ddr_div_con Control DDR divider frequency 2'b00: clk_ddr_src:clk_ddrphy = 1:1 2'b01: clk_ddr_src:clk_ddrphy = 2:1 2'b11: clk_ddr_src:clk_ddrphy = 4:1

**CRU\_CLKSEL27\_CON**

Address: Operational Base + offset (0x00cc)

Internal clock select and divide register27

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:8	RW	0x07	lcdc0_div_con Control LCDC0 clock divider frequency $clk_{lcdc0} = clk_{src}/(lcdc0\_div\_con + 1)$
7:2	RO	0x0	reserved
1:0	RW	0x0	lcdc0_pll_sel Control LCDC0 clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select new pll clock

**CRU\_CLKSEL28\_CON**

Address: Operational Base + offset (0x00d0)

Internal clock select and divide register28

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	edp_24m_sel eDP 24M clock source selection 1'b00: select 27M clock 1'b01: select 24M clock
14:13	RO	0x0	reserved
12:8	RW	0x0f	hclk_vio_div_con VIO AHB clock divider frequency $clk = clk_{src}/(div\_con + 1)$
7:6	RW	0x0	edp_pll_sel eDP clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select new pll clock
5:0	RW	0x03	edp_div_con eDP clock divider frequency $clk = clk_{src}/(div\_con + 1)$

**CRU\_CLKSEL29\_CON**

Address: Operational Base + offset (0x00d4)

Internal clock select and divide register29

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:8	RW	0x07	lc当地1_div_con Control LCD1 clock divider frequency $clk_{lcdc1} = clk_{src} / (lc当地1\_div\_con + 1)$
7:6	RW	0x1	lc当地1_pll_sel Control LCD1 clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select new pll clock
5	RO	0x0	reserved
4	RW	0x0	cif_clkin_inv_sel CIF clkin invert selection 1'b0: normal 1'b1: invert
3	RW	0x0	isp_clkin_inv_sel ISP clkin invert selection 1'b0: normal 1'b1: invert
2	RW	0x0	hsicphy_12m_sel Control HSICPHY 12m clock selection 1'b0: select 12M from OSC 1'b1: select 12M from divout
1:0	RW	0x2	hsicphy_pll_sel Control HSICPHY clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select usbphy 480M clock

**CRU\_CLKSEL30\_CON**

Address: Operational Base + offset (0x00d8)

Internal clock select and divide register30

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RW	0x0	rga_core_clk_pll_sel rga func clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select usbphy pll 480M clock
13	RO	0x0	reserved
12:8	RW	0x00	rga_core_clk_div_con rga func clock divider frequency clk_rga_func = clk_rga_func_src/(rga_core_clk_div_con+1)
7:6	RW	0x0	rga_aclk_pll_sel Control rga AXI clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select usbphy pll 480M clock
5	RO	0x0	reserved
4:0	RW	0x00	rga_aclk_div_con Control rga AXI clock divider frequency aclk_lcdc1=lcdc1_aclk_src/(rga_aclk_div_co n+1)

**CRU\_CLKSEL31\_CON**

Address: Operational Base + offset (0x00dc)

Internal clock select and divide register31

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RW	0x0	vio1_aclk_pll_sel Control VIO1 AXI clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select usbphy pll 480M clock
13	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12:8	RW	0x00	vio1_aclk_div_con Control VIO1 AXI clock divider frequency $aclk\_vio1=vio1\_aclk\_src/(vio1\_aclk\_div\_con+1)$
7:6	RW	0x0	vio0_aclk_pll_sel Control VIO0 AXI clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select usbphy pll 480M clock
5	RO	0x0	reserved
4:0	RW	0x00	vio0_aclk_div_con Control VIO0 AXI clock divider frequency $aclk\_vio0=vio0\_aclk\_src/(vio0\_aclk\_div\_con+1)$

**CRU\_CLKSEL32\_CON**

Address: Operational Base + offset (0x00e0)

Internal clock select and divide register32

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RW	0x0	vdpu_aclk_pll_sel Control VDPU AXI clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select usbphy pll 480M clock
13	RO	0x0	reserved
12:8	RW	0x01	vdpu_aclk_div_con Control VDPU AXI clock divider frequency $aclk\_vdpu=vdpu\_aclk\_src/(vdpu\_aclk\_div\_con+1)$
7:6	RW	0x0	vepu_aclk_pll_sel Control VEPU AXI clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select usbphy pll 480M clock
5	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:0	RW	0x01	<p>vepu_aclk_div_con Control VEPU AXI clock divider frequency <math>aclk_{vepu} = vepu\_aclk\_src / (vepu\_aclk\_div\_con + 1)</math></p>

**CRU\_CLKSEL33\_CON**

Address: Operational Base + offset (0x00e4)

Internal clock select and divide register33

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_mask write mask When HIGH, enable the writing corresponding bit When LOW, don't care the writing corresponding bit</p>
15:13	RO	0x0	reserved
12:8	RW	0x03	<p>alive_pclk_div_con alive apb clock divider frequency alive_pclk <math>=alive\_pclk\_src / (alive\_pclk\_div\_con + 1)</math></p>
7:5	RO	0x0	reserved
4:0	RW	0x03	<p>pmu_pclk_div_con pmu apb clock divider frequency pmu_pclk <math>=pmu\_pclk\_src / (pmu\_pclk\_div\_con + 1)</math></p>

**CRU\_CLKSEL34\_CON**

Address: Operational Base + offset (0x00e8)

Internal clock select and divide register34

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit</p>
15:14	RW	0x2	<p>sdio1_pll_sel Control sdio1 clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select 24MHz</p>
13:8	RW	0x00	<p>sdio1_div_con Control SDIO1 divider frequency <math>clk_{sdio} = general\_pll\_clk / (sdio\_div\_con + 1)</math></p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:6	RW	0x0	gpu_aclk_pll_sel Control GPU AXI clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select usbphy pll 480M clock 2'b11: select new pll clock
5	RO	0x0	reserved
4:0	RW	0x00	gpu_aclk_div_con Control GPU AXI clock divider frequency $aclk_{gpu} = gpu\_aclk\_src / (gpu\_aclk\_div\_con + 1)$

**CRU\_CLKSEL35\_CON**

Address: Operational Base + offset (0x00ec)

Internal clock select and divide register35

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RW	0x0	tspout_clk_pll_sel Control tspout clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select new pll clock 2'b11: select 27MHz IO input
13	RO	0x0	reserved
12:8	RW	0x03	tspout_clk_div_con Control tspout clock divider frequency $clk = clk\_src / (clk\_div\_con + 1)$
7:6	RW	0x0	tsp_clk_pll_sel Control tsp clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select new pll clock
5	RO	0x0	reserved
4:0	RW	0x03	tsp_clk_div_con Control tsp clock divider frequency $clk = clk\_src / (clk\_div\_con + 1)$

**CRU\_CLKSEL36\_CON**

Address: Operational Base + offset (0x00f0)

Internal clock select and divide register36

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RO	0x0	reserved
14:12	RW	0x0	clk_core3_div_con Control clk_core3 clock divider frequency $clk=clk\_src/(clk\_div\_con+1)$
11	RO	0x0	reserved
10:8	RW	0x0	clk_core2_div_con Control clk_core2 clock divider frequency $clk=clk\_src/(clk\_div\_con+1)$
7	RO	0x0	reserved
6:4	RW	0x0	clk_core1_div_con Control clk_core1 clock divider frequency $clk=clk\_src/(clk\_div\_con+1)$
3	RO	0x0	reserved
2:0	RW	0x0	clk_core0_div_con Control clk_core0 clock divider frequency $clk=clk\_src/(clk\_div\_con+1)$

**CRU\_CLKSEL37\_CON**

Address: Operational Base + offset (0x00f4)

Internal clock select and divide register37

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved
13:9	RW	0x0f	pclk_core_dbg_div_con Control core dbg APB bus clock divider frequency $clk=clk\_src/(clk\_div\_con+1)$
8:4	RW	0x0f	atclk_core_div_con Control core ATB BUS clock divider frequency $clk=clk\_src/(clk\_div\_con+1)$
3	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:0	RW	0x3	clk_l2ram_div_con Control clk_l2ram clock divider frequency $clk=clk\_src/(clk\_div\_con+1)$

**CRU\_CLKSEL38\_CON**

Address: Operational Base + offset (0x00f8)

Internal clock select and divide register38

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	nandc1_clk_pll_sel Control nandc1 clock PLL source selection 1'b0: select codec pll clock 1'b1: select general pll clock
14:13	RO	0x0	reserved
12:8	RW	0x03	nandc1_clk_div_con Control nandc1 clock divider frequency $clk\_nandc=nandc\_clk\_src/(nandc\_clk\_div\_con+1)$
7	RW	0x0	nandc0_clk_pll_sel Control nandc0 clock PLL source selection 1'b0: select codec pll clock 1'b1: select general pll clock
6:5	RO	0x0	reserved
4:0	RW	0x03	nandc0_clk_div_con Control nandc0 clock divider frequency $clk\_nandc=nandc\_clk\_src/(nandc\_clk\_div\_con+1)$

**CRU\_CLKSEL39\_CON**

Address: Operational Base + offset (0x00fc)

Internal clock select and divide register39

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:14	RW	0x0	aclk_hevc_pll_sel HEVC AXI clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select new pll clock
13	RO	0x0	reserved
12:8	RW	0x00	aclk_hevc_div_con HEVC AXI clock divider frequency $clk=clk\_src/(clk\_div\_con+1)$
7	RW	0x0	spi2_pll_sel Control spi2 clock PLL source selection 1'b0: select codec pll clock 1'b1: select general pll clock
6:0	RW	0x07	spi2_div_con Control SPI2 clock divider frequency $clk=clk\_src/(div\_con+1)$

**CRU\_CLKSEL40\_CON**

Address: Operational Base + offset (0x0100)

Internal clock select and divide register40

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved
13:12	RW	0x0	hclk_hevc_div_con HEVC AHB clock divider frequency $clk=clk\_src/(clk\_div\_con+1)$
11:10	RO	0x0	reserved
9:8	RW	0x2	spdif_8ch_clk_sel Control SPDIF 8ch clock work frequency selection 2'b00: select divider output from pll divider 2'b01: select divider output from fraction divider 2'b10: select 12MHz from osc input
7	RO	0x0	reserved
6:0	RW	0x00	spdif_8ch_pll_div_con Control SPDIF 8ch PLL output divider frequency $spdif\_div\_clk=spdif\_div\_src/(spdif\_pll\_div\_con+1)$

**CRU\_CLKSEL41\_CON**

Address: Operational Base + offset (0x0104)

Internal clock select and divide register41

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x0bb8ea60	spdif_8ch_frac_factor Control SPDIF 8ch fraction divider frequency High 16-bit for numerator Low 16-bit for denominator

**CRU\_CLKSEL42\_CON**

Address: Operational Base + offset (0x0108)

Internal clock select and divide register42

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RW	0x0	clk_hevc_core_pll_sel HEVC CORE clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select new pll clock
13	RO	0x0	reserved
12:8	RW	0x00	clk_hevc_core_div_con HEVC CORE clock divider frequency $clk=clk\_src/(clk\_div\_con+1)$
7:6	RW	0x0	clk_hevc_cabac_pll_sel HEVC CABAC clock PLL source selection 2'b00: select codec pll clock 2'b01: select general pll clock 2'b10: select new pll clock
5	RO	0x0	reserved
4:0	RW	0x00	clk_hevc_cabac_div_con HEVC CABAC clock divider frequency $clk=clk\_src/(clk\_div\_con+1)$

**CRU\_CLKGATE0\_CON**

Address: Operational Base + offset (0x0160)

Internal clock gating control register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:13	RO	0x0	reserved
12	RW	0x0	clk_acc_efuse_gate_en acc efuse clock disable. When HIGH, disable clock
11	RW	0x0	pd_bus_cpll_clk_gate_en pd_bus clock CPLL path clock disable. When HIGH, disable clock
10	RW	0x0	pd_bus_gpll_clk_gate_en pd_bus clock GPLL path clock disable. When HIGH, disable clock
9	RW	0x0	ddr_gpll_clk_gate_en DDR clock GPLL path clock disable. When HIGH, disable clock
8	RW	0x0	ddr_dpll_clk_gate_en DDR clock DPLL path clock disable. When HIGH, disable clock
7	RW	0x0	aclk_bus_2pmu_gate_en pd_bus AXI clock to pd_pmu clock disable. When HIGH, disable clock
6	RO	0x0	reserved
5	RW	0x0	pclk_bus_gate_en pd_bus APB clock(pclk_cpu_pre) disable. When HIGH, disable clock
4	RW	0x0	hclk_bus_gate_en pd_bus AHB clock disable. When HIGH, disable clock
3	RW	0x0	aclk_bus_gate_en pd_bus AXI clock disable. When HIGH, disable clock
2	RW	0x0	core_gpll_clk_gate_en CORE clock GPLL path clock disable. When HIGH, disable clock
1	RW	0x0	core_apll_clk_gate_en CORE clock APLL path clock disable. When HIGH, disable clock
0	RO	0x0	reserved

**CRU\_CLKGATE1\_CON**

Address: Operational Base + offset (0x0164)

## Internal clock gating control register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	clk_uart3_frac_src_gate_en UART3 fraction divider source clock disable. When HIGH, disable clock
14	RW	0x0	clk_uart3_src_gate_en UART3 source clock disable. When HIGH, disable clock
13	RW	0x0	clk_uart2_frac_src_gate_en UART2 fraction divider source clock disable. When HIGH, disable clock
12	RW	0x0	clk_uart2_src_gate_en UART2 source clock disable. When HIGH, disable clock
11	RW	0x0	clk_uart1_frac_src_gate_en UART1 fraction divider source clock disable. When HIGH, disable clock
10	RW	0x0	clk_uart1_src_gate_en UART1 source clock disable. When HIGH, disable clock
9	RW	0x0	clk_uart0_frac_src_gate_en UART0 fraction divider source clock disable. When HIGH, disable clock
8	RW	0x0	clk_uart0_src_gate_en UART0 source clock disable. When HIGH, disable clock
7:6	RO	0x0	reserved
5	RW	0x0	clk_timer5_gate_en Timer5 clock(clk_timer5) disable. When HIGH, disable clock
4	RW	0x0	clk_timer4_gate_en Timer4 clock(clk_timer4) disable. When HIGH, disable clock
3	RW	0x0	clk_timer3_gate_en Timer3 clock(clk_timer3) disable. When HIGH, disable clock
2	RW	0x0	clk_timer2_gate_en Timer2 clock(clk_timer2) disable. When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	clk_timer1_gate_en Timer1 clock(clk_timer1) disable. When HIGH, disable clock
0	RW	0x0	clk_timer0_gate_en Timer0 clock(clk_timer0) disable. When HIGH, disable clock

**CRU\_CLKGATE2\_CON**

Address: Operational Base + offset (0x0168)

Internal clock gating control register2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved
13	RW	0x0	clk_uart4_frac_src_gate_en UART4 fraction divider source clock disable. When HIGH, disable clock
12	RW	0x0	clk_uar4_src_gate_en UART4 source clock disable. When HIGH, disable clock
11	RW	0x0	clk_spi2_src_gate_en SPI2 source clock disable. When HIGH, disable clock
10	RW	0x0	clk_spi1_src_gate_en SPI1 source clock disable. When HIGH, disable clock
9	RW	0x0	clk_spi0_src_gate_en SPI0 source clock disable. When HIGH, disable clock
8	RW	0x0	clk_saradc_src_gate_en SARADC source clock disable. When HIGH, disable clock
7	RW	0x0	clk_tsadc_src_gate_en TSADC source clock disable. When HIGH, disable clock
6	RW	0x0	clk_hsadc_src_gate_en Field0000 Abstract When HIGH, disable clock
5	RW	0x0	clk_mac_src_gate_en MAC source clock disable. When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RO	0x0	reserved
3	RW	0x0	pclk_periph_gate_en PERIPH system APB clock(pclk_periph) disable. When HIGH, disable clock
2	RW	0x0	hclk_periph_gate_en PERIPH system AHB clock(hclk_periph) disable. When HIGH, disable clock
1	RW	0x0	aclk_periph_gate_en PERIPH system AXI clock(aclk_periph) disable. When HIGH, disable clock
0	RW	0x0	clk_periph_src_gate_en PERIPH system source clock disable. When HIGH, disable clock

**CRU\_CLKGATE3\_CON**

Address: Operational Base + offset (0x016c)

Internal clock gating control register3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	clk_isp_jpeg_gate_en ISP jpeg source clock disable. When HIGH, disable clock
14	RW	0x0	clk_isp_gate_en ISP clock clock disable. When HIGH, disable clock
13	RW	0x0	clk_edp_gate_en eDP clock clock disable. When HIGH, disable clock
12	RW	0x0	clk_edp_24m_gate_en eDP 24M ref clock clock disable. When HIGH, disable clock
11	RW	0x0	aclk_vdpu_src_gate_en VDPU AXI source clock disable. When HIGH, disable clock
10	RW	0x0	hclk_vpu_gate_en VPU AHB source clock disable. When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	aclk_vepu_src_gate_en VEPU AXI source clock disable. When HIGH, disable clock
8	RO	0x0	reserved
7	RW	0x0	clk_cif_out_gate_en CIF output clock disable. When HIGH, disable clock
6	RW	0x0	hsicphy_gate_en HSICPHY clock disable. When HIGH, disable clock
5	RW	0x0	aclk_rga_src_gate_en RGA AXI souce clock disable. When HIGH, disable clock
4	RW	0x0	clk_rga_core_src_gate_en RGA func souce clock disable. When HIGH, disable clock
3	RW	0x0	dclk_lcdc1_src_gate_en LCDC1 DCLK source clock disable. When HIGH, disable clock
2	RW	0x0	aclk_lcdc1_src_gate_en LCDC1 AXI source clock disable. When HIGH, disable clock
1	RW	0x0	dclk_lcdc0_src_gate_en LCDC0 DCLK source clock disable. When HIGH, disable clock
0	RW	0x0	aclk_lcdc0_src_gate_en LCDC0 AXI source clock disable. When HIGH, disable clock

**CRU\_CLKGATE4\_CON**

Address: Operational Base + offset (0x0170)

Internal clock gating control register4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	testclk_gate_en Test output clock disable When HIGH, disable clock
14	RW	0x0	clk_jtag_gate_en JTAG clock disable. When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13	RW	0x0	clk_ddrphy1_gate_en DDRPHY1 clock disable. When HIGH, disable clock
12	RW	0x0	clk_ddrphy0_gate_en DDRPHY0 clock disable. When HIGH, disable clock
11	RW	0x0	clk_tspout_gate_en TSP output clock disable. When HIGH, disable clock
10	RW	0x0	clk_tsp_gate_en TSP clock disable. When HIGH, disable clock
9	RW	0x0	clk_spdif_8ch_gate_en SPDIF 8ch clock disable. When HIGH, disable clock
8	RW	0x0	clk_spdif_8ch_frac_src_gate_en SPDIF 8ch fraction divider source clock disable. When HIGH, disable clock
7	RW	0x0	clk_spdif_8ch_src_gate_en SPDIF 8ch source clock disable. When HIGH, disable clock
6	RW	0x0	clk_spdif_gate_en SPDIF clock disable. When HIGH, disable clock
5	RW	0x0	clk_spdif_frac_src_gate_en SPDIF fraction divider source clock disable. When HIGH, disable clock
4	RW	0x0	clk_spdif_src_gate_en SPDIF source clock disable. When HIGH, disable clock
3	RW	0x0	clk_i2s0_gate_en I2S clock disable. When HIGH, disable clock
2	RW	0x0	clk_i2s0_frac_src_gate_en I2S fraction divider source clock disable. When HIGH, disable clock
1	RW	0x0	clk_i2s0_src_gate_en I2S source clock disable. When HIGH, disable clock
0	RW	0x0	clk_i2s0_out_gate_en I2S output clock disable. When HIGH, disable clock

**CRU\_CLKGATE5\_CON**

Address: Operational Base + offset (0x0174)

Internal clock gating control register5

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	clk_mipidsi_24m_gate_en mipi dsi 24M clock disable. When HIGH, disable clock
14	RW	0x0	clk_usbphy480m_gate_en usbphy480M clock disable. When HIGH, disable clock
13	RW	0x0	ps2c_clk_gate_en PS2 controlor clock disable. When HIGH, disable clock
12	RW	0x0	hdmi_hdcp_clk_gate_en HDMI HDCP clock disable. When HIGH, disable clock
11	RW	0x0	hdmi_cec_clk_gate_en HDMI CEC clock disable. When HIGH, disable clock
10	RW	0x0	clk_pvtm_gpu_gate_en pd_gpu PVTM clock disable. When HIGH, disable clock
9	RW	0x0	clk_pvtm_core_gate_en pd_core PVTM clock disable. When HIGH, disable clock
8	RW	0x0	pclk_pmu_gate_en pd_pmu APB bus clock disable. When HIGH, disable clock
7	RW	0x0	clk_gpu_gate_en gpu clock disable. When HIGH, disable clock
6	RW	0x0	clk_nandc1_gate_en nandc1 clock disable. When HIGH, disable clock
5	RW	0x0	clk_nandc0_gate_en nandc0 clock disable. When HIGH, disable clock
4	RW	0x0	clk_crypto_gate_en crypto clock disable. When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	clk_mac_refout_gate_en MAC ref output clock clock disable. When HIGH, disable clock
2	RW	0x0	clk_mac_ref_gate_en MAC ref clock clock disable. When HIGH, disable clock
1	RW	0x0	clk_mac_tx_gate_en MAC tx clock clock disable. When HIGH, disable clock
0	RW	0x0	clk_mac_rx_gate_en MAC rx clock clock disable. When HIGH, disable clock

**CRU\_CLKGATE6\_CON**

Address: Operational Base + offset (0x0178)

Internal clock gating control register6

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	pclk_i2c4_gate_en I2C4 APB clock disable. When HIGH, disable clock
14	RW	0x0	pclk_i2c3_gate_en I2C3 APB clock disable. When HIGH, disable clock
13	RW	0x0	pclk_i2c2_gate_en I2C2 APB clock disable. When HIGH, disable clock
12	RW	0x0	pclk_uart_exp_gate_en UART_exp APB clock disable. When HIGH, disable clock
11	RW	0x0	pclk_uart_gps_gate_en UART_gps APB clock disable. When HIGH, disable clock
10	RO	0x0	reserved
9	RW	0x0	pclk_uart_bb_gate_en UART_bb APB clock disable. When HIGH, disable clock
8	RW	0x0	pclk_uart_bt_gate_en UART_bt APB clock disable. When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	pclk_ps2c0_gate_en PS2C0 APB clock disable. When HIGH, disable clock
6	RW	0x0	pclk_spi2_gate_en SPI2 APB clock disable. When HIGH, disable clock
5	RW	0x0	pclk_spi1_gate_en SPI1 APB clock disable. When HIGH, disable clock
4	RW	0x0	pclk_spi0_gate_en SPI0 APB clock disable. When HIGH, disable clock
3	RW	0x0	aclk_dmac_peri_gate_en DMAC peri AXI clock disable. When HIGH, disable clock
2	RW	0x0	aclk_peri_axi_matrix_gate_en Peripheral matrix axi clock disable. When HIGH, disable clock
1	RW	0x0	pclk_peri_axi_matrix_gate_en Peripheral matrix apb clock disable. When HIGH, disable clock
0	RW	0x0	hclk_peri_matrix_gate_en Peripheral matrix ahb clock disable. When HIGH, disable clock

**CRU\_CLKGATE7\_CON**

Address: Operational Base + offset (0x017c)

Internal clock gating control register7

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	hclk_nand1_gate_en NAND1 AHB clock disable. When HIGH, disable clock
14	RW	0x0	hclk_nand0_gate_en NAND0 AHB clock disable. When HIGH, disable clock
13	RW	0x0	hclk_mmc_peri_gate_en arbiter in peri_ahb_mmc module AHB clock disable. When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x0	hclk_emem_peri_gate_en arbiter in peri_ahb_emem module AHB clock disable. When HIGH, disable clock
11	RW	0x0	aclk_peri_niu_gate_en NIU in peripheral power domain AXI clock disable. When HIGH, disable clock
10	RW	0x0	hclk_peri_ahb_arbi_gate_en AHB arbiter in peripheral power domain AHB clock disable. When HIGH, disable clock
9	RW	0x0	hclk_usb_peri_gate_en USB arbiter AHB clock disable. When HIGH, disable clock
8	RW	0x0	hclk_hsic_gate_en HSIC AHB clock disable. When HIGH, disable clock
7	RW	0x0	hclk_host1_gate_en HOST1 AHB clock disable. Field0000 Description
6	RW	0x0	hclk_host0_gate_en HOST0 AHB clock disable. Field0000 Description
5	RW	0x0	pmu_hclk_otg0_gate_en USB OTG PMU AHB clock disable. When HIGH, disable clock
4	RW	0x0	hclk_otg0_gate_en USB OTG AHB clock disable. When HIGH, disable clock
3	RW	0x0	pclk_sim_gate_en SIM APB clock disable. When HIGH, disable clock
2	RW	0x0	pclk_tsadc_gate_en TSADC APB clock disable. When HIGH, disable clock
1	RW	0x0	pclk_saradc_gate_en SARADC APB clock disable. When HIGH, disable clock
0	RW	0x0	pclk_i2c5_gate_en I2C5 APB clock disable. When HIGH, disable clock

**CRU\_CLKGATE8\_CON**

Address: Operational Base + offset (0x0180)

## Internal clock gating control register8

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:13	RO	0x0	reserved
12	RW	0x0	aclk_peri_mmu_gate_en PERI_MMU aclk clock disable. When HIGH, disable clock
11	RW	0x0	clk_27m_tsp_gate_en 27M_TSP clock disable. When HIGH, disable clock
10	RW	0x0	clk_hsadc_1_tsp_gate_en HSADC_1_TSP clock disable. When HIGH, disable clock
9	RW	0x0	clk_hsadc_0_tsp_gate_en HSADC_0_TSP clock disable. When HIGH, disable clock
8	RW	0x0	hclk_tsp_gate_en TSP AHB clock disable. When HIGH, disable clock
7	RW	0x0	hclk_hsadc_gate_en HSADC AHB clock disable. When HIGH, disable clock
6	RW	0x0	hclk_emmc_gate_en EMMC AHB clock disable. When HIGH, disable clock
5	RW	0x0	hclk_sdio1_gate_en SDIO1 AHB clock disable. When HIGH, disable clock
4	RW	0x0	hclk_sdio0_gate_en SDIO0 AHB clock disable. When HIGH, disable clock
3	RW	0x0	hclk_sdmmc_gate_en SDMMC AHB clock disable. When HIGH, disable clock
2	RW	0x0	hclk_gps_gate_en GPS hclk clock disable. When HIGH, disable clock
1	RW	0x0	pclk_gmac_gate_en GMAC pclk clock disable. When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	aclk_gmac_gate_en GMAC aclk clock disable. When HIGH, disable clock

**CRU\_CLKGATE9\_CON**

Address: Operational Base + offset (0x0184)

Internal clock gating control register9

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:2	RO	0x0	reserved
1	RW	0x0	hclk_video_clock_en VIDEO AHB clock disable. When HIGH, disable clock
0	RW	0x0	aclk_video_gate_en VIDEO AXI clock disable. When HIGH, disable clock

**CRU\_CLKGATE10\_CON**

Address: Operational Base + offset (0x0188)

Internal clock gating control register10

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	pclk_publ0_gate_en DDR0 PUBL apb clock disable When HIGH, disable clock
14	RW	0x0	pclk_ddrupctl0_gate_en DDDRUPCTL0 apb clock disable When HIGH, disable clock
13	RW	0x0	aclk_strc_sys_gate_en aclk_strc_sys (CPU Structure system) clock disable. When HIGH, disable clock
12	RW	0x0	aclk_dmac_bus_gate_en DMAC_BUS aclk clock disable. When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x0	hclk_spdif_8ch_gate_en hclk_spdif_8ch clock disable. When HIGH, disable clock
10	RW	0x0	hclk_spdif_gate_en hclk_spdif clock disable. When HIGH, disable clock
9	RW	0x0	hclk_rom_gate_en hclk_rom clock disable. When HIGH, disable clock
8	RW	0x0	hclk_i2s_8ch_gate_en hclk_i2s_8ch AHB clock disable. When HIGH, disable clock
7	RW	0x0	clk_intmem2_gate_en intmem2 clock disable. When HIGH, disable clock
6	RW	0x0	clk_intmem1_gate_en intmem1 clock disable. When HIGH, disable clock
5	RW	0x0	clk_intmem0_gate_en intmem0 clock disable. When HIGH, disable clock
4	RW	0x0	aclk_intmem_gate_en intmem axi clock disable. When HIGH, disable clock
3	RW	0x0	pclk_i2c1_gate_en pclk_i2c1 disable. When HIGH, disable clock
2	RW	0x0	pclk_i2c0_gate_en pclk_i2c0 disable. When HIGH, disable clock
1	RW	0x0	pclk_timer_gate_en pclk_timer disable. When HIGH, disable clock
0	RW	0x0	pclk_pwm_gate_en pclk_pwm disable. When HIGH, disable clock

**CRU\_CLKGATE11\_CON**

Address: Operational Base + offset (0x018c)

Internal clock gating control register11

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:12	RO	0x0	reserved
11	RW	0x0	pclk_rkpwm_gate_en pclk_rkpwm disable. When HIGH, disable clock
10	RW	0x0	pclk_efuse_256_gate_en EFUSE256 APB clock disable. When HIGH, disable clock
9	RW	0x0	pclk_uart_dbg_gate_en UART_DBG APB clock disable. When HIGH, disable clock
8	RW	0x0	aclk_ccp_gate_en CCP aclk clock disable. When HIGH, disable clock
7	RW	0x0	hclk_crypto_gate_en CRYPTO sclk clock disable. When HIGH, disable clock
6	RW	0x0	aclk_crypto_gate_en CRYPTO mclk clock disable. When HIGH, disable clock
5	RW	0x0	nclk_ddrupctl1_gate_en DDR Controller PHY clock disable. When HIGH, disable clock
4	RW	0x0	nclk_ddrupctl0_gate_en DDR Controller PHY clock disable. When HIGH, disable clock
3	RW	0x0	pclk_tzpc_gate_en TZPC APB clock disable. When HIGH, disable clock
2	RW	0x0	pclk_efuse_1024_gate_en EFUSE1024 APB clock disable. When HIGH, disable clock
1	RW	0x0	pclk_publ1_gate_en DDR1 PUBL apb clock disable When HIGH, disable clock
0	RW	0x0	pclk_ddrupctl1_gate_en DDDRUPCTL1 apb clock disable When HIGH, disable clock

**CRU\_CLKGATE12\_CON**

Address: Operational Base + offset (0x0190)

Internal clock gating control register12

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:12	RO	0x0	reserved
11	RW	0x0	pclk_core_niu_gate_en core NIU APB bus clock disable. When HIGH, disable clock
10	RW	0x0	cs_dbg_clk_gate_en coresight debug clock disable. When HIGH, disable clock
9	RW	0x0	dbg_core_clk_gate_en core debug clock disable. When HIGH, disable clock
8	RW	0x0	dbg_src_clk_gate_en Debug source clock disable. When HIGH, disable clock
7	RW	0x0	atclk_core_gate_en core ATB bus clock disable. When HIGH, disable clock
6	RW	0x0	aclk_mp_gate_en core MP AXI bus clock disable. When HIGH, disable clock
5	RW	0x0	aclk_core_m0_gate_en CORE m0 AXI bus clock disable. When HIGH, disable clock
4	RW	0x0	l2_ram_clk_gate_en L2 RAM clock disable. When HIGH, disable clock
3	RW	0x0	core3_clk_gate_en core3 clock disable. When HIGH, disable clock
2	RW	0x0	core2_clk_gate_en core2 clock disable. When HIGH, disable clock
1	RW	0x0	coer1_clk_gate_en core1 clock disable. When HIGH, disable clock
0	RW	0x0	core0_clk_gate_en core0 clock disable. When HIGH, disable clock

**CRU\_CLKGATE13\_CON**

Address: Operational Base + offset (0x0194)

Internal clock gating control register13

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	clk_hevc_core_gate_en HEVC CORE clock disable. When HIGH, disable clock
14	RW	0x0	clk_hevc_cabac_gate_en HEVC cabac clock disable. When HIGH, disable clock
13	RW	0x0	aclk_hevc_gate_en HEVC AXI clock disable. When HIGH, disable clock
12	RW	0x0	clk_wifi_gate_en wifi/gps/bt 3in1 16.384M clock disable. When HIGH, disable clock
11	RW	0x0	clk_lcdc_pwm1_gate_en lcdc_pwm1 clock disable. When HIGH, disable clock
10	RW	0x0	clk_lcdc_pwm0_gate_en lcdc_pwm0 clock disable. When HIGH, disable clock
9	RW	0x0	clk_hsic_12m_gate_en HSIC 12MHz clock disable. When HIGH, disable clock
8	RW	0x0	clk_c2c_host_gate_en C2C HOST clock disable. When HIGH, disable clock
7	RW	0x0	clk_otg_adp_gate_en OTG adp clock disable. When HIGH, disable clock
6	RW	0x0	clk_otgphy2_gate_en OTGPHY2 clock(clk_otgphy2) disable. When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	clk_otgphy1_gate_en OTGPHY1 clock(clk_otgphy1) disable. When HIGH, disable clock
4	RW	0x0	clk_otgphy0_gate_en OTGPHY0 clock(clk_otgphy0) disable. When HIGH, disable clock
3	RW	0x0	clk_emmc_src_gate_en EMMC source clock disable. When HIGH, disable clock
2	RW	0x0	clk_sdio1_src_gate_en SDIO1 source clock disable. When HIGH, disable clock
1	RW	0x0	clk_sdio0_src_gate_en SDIO0 source clock disable. When HIGH, disable clock
0	RW	0x0	clk_mmc0_src_gate_en SDMMC0 source clock disable. When HIGH, disable clock

**CRU\_CLKGATE14\_CON**

Address: Operational Base + offset (0x0198)

Internal clock gating control register14

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:13	RO	0x0	reserved
12	RW	0x0	pclk_alive_niu_gate_en ALIVE_NIU pclk disable When HIGH, disable clock
11	RW	0x0	pclk_grf_gate_en GRF pclk disable When HIGH, disable clock
10:9	RO	0x0	reserved
8	RW	0x0	pclk_gpio8_gate_en GPIO8 pclk disable When HIGH, disable clock
7	RW	0x0	pclk_gpio7_gate_en GPIO7 pclk disable When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RW	0x0	pclk_gpio6_gate_en GPIO6 pclk disable When HIGH, disable clock
5	RW	0x0	pclk_gpio5_gate_en GPIO5 pclk disable When HIGH, disable clock
4	RW	0x0	pclk_gpio4_gate_en GPIO4 pclk disable When HIGH, disable clock
3	RW	0x0	pclk_gpio3_gate_en GPIO3 pclk disable When HIGH, disable clock
2	RW	0x0	pclk_gpio2_gate_en GPIO2 pclk disable When HIGH, disable clock
1	RW	0x0	pclk_gpio1_gate_en GPIO1 pclk disable When HIGH, disable clock
0	RO	0x0	reserved

**CRU\_CLKGATE15\_CON**

Address: Operational Base + offset (0x019c)

Internal clock gating control register15

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	hclk_vip_gate_en VIP hclk disable When HIGH, disable clock
14	RW	0x0	aclk_vip_gate_en VIP aclk disable When HIGH, disable clock
13	RW	0x0	aclk_vio2_noc_gate_en VIO2_NOC aclk disable When HIGH, disable clock
12	RW	0x0	aclk_vio1_noc_gate_en VIO1_NOC aclk disable When HIGH, disable clock
11	RW	0x0	aclk_vio0_noc_gate_en VIO0_NOC aclk disable When HIGH, disable clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
10	RW	0x0	hclk_vio_noc_gate_en VIO_NOC hclk disable When HIGH, disable clock
9	RW	0x0	hclk_vio_ahb_arbi_gate_en VIO_AHB_ARBI hclk disable When HIGH, disable clock
8	RW	0x0	hclk_lcdc1_gate_en LCDC1 hclk disable When HIGH, disable clock
7	RW	0x0	aclk_lcdc1_gate_en LCDC1 aclk disable When HIGH, disable clock
6	RW	0x0	hclk_lcdc0_gate_en LCDC0 hclk disable When HIGH, disable clock
5	RW	0x0	aclk_lcdc0_gate_en LCDC0 aclk disable When HIGH, disable clock
4	RW	0x0	aclk_lcdc_iep_gate_en LCDC_IEP aclk disable When HIGH, disable clock
3	RW	0x0	hclk_iep_gate_en IEP hclk disable When HIGH, disable clock
2	RW	0x0	aclk_iep_gate_en IEP aclk disable When HIGH, disable clock
1	RW	0x0	hclk_rga_gate_en RGA hclk disable When HIGH, disable clock
0	RW	0x0	aclk_rga_gate_en RGA aclk disable When HIGH, disable clock

**CRU\_CLKGATE16\_CON**

Address: Operational Base + offset (0x01a0)

Internal clock gating control register16

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:12	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x0	pclk_vio2_h2p_gate_en VIO2_H2P pclk disable When HIGH, disable clock
10	RW	0x0	hclk_vio2_h2p_gate_en VIO2_H2P hclk disable When HIGH, disable clock
9	RW	0x0	pclk_hdmi_ctrl_gate_en HDMI_CTRL pclk disable When HIGH, disable clock
8	RW	0x0	pclk_edp_ctrl_gate_en EDP_CTRL pclk disable When HIGH, disable clock
7	RW	0x0	pclk_lvds_phy_gate_en LVDS_PHY pclk disable When HIGH, disable clock
6	RW	0x0	pclk_mipi_csi_gate_en MIPI_CSI pclk disable When HIGH, disable clock
5	RW	0x0	pclk_mipi_dsi1_gate_en MIPI_DSI1 pclk disable When HIGH, disable clock
4	RW	0x0	pclk_mipi_dsi0_gate_en MIPI_DSI0 pclk disable When HIGH, disable clock
3	RW	0x0	pclk_in_isp_gate_en ISP pclkin disable When HIGH, disable clock
2	RW	0x0	aclk_isp_gate_en ISP aclk disable When HIGH, disable clock
1	RW	0x0	hclk_isp_gate_en ISP hclk disable When HIGH, disable clock
0	RW	0x0	pclk_in_vip_gate_en VIP pclkin disable When HIGH, disable clock

**CRU\_CLKGATE17\_CON**

Address: Operational Base + offset (0x01a4)

Internal clock gating control register17

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:5	RO	0x0	reserved
4	RW	0x0	pclk_gpio0_gate_en GPIO0 pclk disable When HIGH, disable clock
3	RW	0x0	pclk_sgrf_gate_en SGRF pclk disable When HIGH, disable clock
2	RW	0x0	pclk_pmu_noc_gate_en PMU_NOC pclk disable When HIGH, disable clock
1	RW	0x0	pclk_intmem1_gate_en INTMEM1 pclk disable When HIGH, disable clock
0	RW	0x0	pclk_pmu_gate_en PMU pclk disable When HIGH, disable clock

**CRU\_CLKGATE18\_CON**

Address: Operational Base + offset (0x01a8)

Internal clock gating control register18

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:1	RO	0x0	reserved
0	RW	0x0	aclk_gpu_gate_en GPU aclk disable When HIGH, disable clock

**CRU\_GLB\_SRST\_FST\_VALUE**

Address: Operational Base + offset (0x01b0)

The first global software reset config value

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	glb_srst_fst_value The first global software reset config value If config 0xfd9, it will generate first global software reset.

**CRU\_GLB\_SRST SND\_VALUE**

Address: Operational Base + offset (0x01b4)

The second global software reset config value

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	glb_srst_snd_value The second global software reset config value If config 0xe8, it will generate second global software reset.

**CRU\_SOFTRST0\_CON**

Address: Operational Base + offset (0x01b8)

Internal software reset control register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	core3_dbg_srstn_req Core3 CPU debug software reset request. When HIGH, reset relative logic
14	RW	0x0	core2_dbg_srstn_req Core2 CPU debug software reset request. When HIGH, reset relative logic
13	RW	0x0	core1_dbg_srstn_req Core1 CPU debug software reset request. When HIGH, reset relative logic
12	RW	0x0	core0_dbg_srstn_req Core0 CPU debug software reset request. When HIGH, reset relative logic
11	RW	0x0	topdbg_srstn_req CPU top debug software reset request. When HIGH, reset relative logic
10	RW	0x0	I2C_srstn_req L2 controller software reset request. When HIGH, reset relative logic

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	pd_bus_str_sys_asrstn_req PD BUS NOC AXI software reset request. When HIGH, reset relative logic
8	RW	0x0	pd_core_str_sys_asrstn_req PD CORE NOC AXI software reset request. When HIGH, reset relative logic
7	RW	0x0	core3_po_srstn_req Core3 CPU PO software reset request. When HIGH, reset relative logic
6	RW	0x0	core2_po_srstn_req Core2 CPU PO software reset request. When HIGH, reset relative logic
5	RW	0x0	core1_po_srstn_req Core1 CPU PO software reset request. When HIGH, reset relative logic
4	RWSC	0x0	core0_po_srstn_req Core0 CPU PO software reset request. When HIGH, reset relative logic
3	RW	0x0	core3_srstn_req Core3 CPU software reset request. When HIGH, reset relative logic
2	RW	0x0	core2_srstn_req Core2 CPU software reset request. When HIGH, reset relative logic
1	RW	0x0	core1_srstn_req Core1 CPU software reset request. When HIGH, reset relative logic
0	RWSC	0x0	core0_srstn_req Core0 CPU software reset request. When HIGH, reset relative logic

**CRU\_SOFTRST1\_CON**

Address: Operational Base + offset (0x01bc)

Internal software reset control register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	efuse_psrstn_req EFUSE APB software reset request. When HIGH, reset relative logic

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14	RW	0x0	timer5_srstn_req Timer5 software reset request. When HIGH, reset relative logic
13	RW	0x0	timer4_srstn_req Timer4 software reset request. When HIGH, reset relative logic
12	RW	0x0	timer3_srstn_req Timer3 software reset request. When HIGH, reset relative logic
11	RW	0x0	timer2_srstn_req Timer2 software reset request. When HIGH, reset relative logic
10	RW	0x0	timer1_srstn_req Timer1 software reset request. When HIGH, reset relative logic
9	RW	0x0	timer0_srstn_req Timer0 software reset request. When HIGH, reset relative logic
8	RW	0x0	spdif_srstn_req SPDIF software reset request. When HIGH, reset relative logic
7	RW	0x0	i2s_srstn_req I2S software reset request. When HIGH, reset relative logic
6	RW	0x0	timer_psrstn_req Timer APB software reset request. When HIGH, reset relative logic
5	RW	0x0	spdif_8ch_srstn_req SPDIF 8ch software reset request. When HIGH, reset relative logic
4	RW	0x0	rom_srstn_req ROM software reset request. When HIGH, reset relative logic
3	RW	0x0	intmem_srstn_req Internal memory software reset request. When HIGH, reset relative logic
2	RW	0x0	dma1_srstn_req DMA1 software reset request. When HIGH, reset relative logic
1	RW	0x0	efuse_256bit_psrstn_req 256bit EFUSE APB software reset request. When HIGH, reset relative logic

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	pd_bus_ahb_arbitor_srstn_req pd_bus ahb arbitor software reset request. pd_cpu AHB arbitor reset control When HIGH, reset relative logic

**CRU\_SOFTRST2\_CON**

Address: Operational Base + offset (0x01c0)

Internal software reset control register2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	i2c5_srstn_req I2C5 software reset request. When HIGH, reset relative logic
14	RW	0x0	i2c4_srstn_req I2C4 software reset request. When HIGH, reset relative logic
13	RW	0x0	i2c3_srstn_req I2C3 software reset request. When HIGH, reset relative logic
12	RW	0x0	i2c2_srstn_req I2C2 software reset request. When HIGH, reset relative logic
11	RW	0x0	i2c1_srstn_req I2C1 software reset request. When HIGH, reset relative logic
10	RW	0x0	i2c0_srstn_req I2C0 software reset request. When HIGH, reset relative logic
9	RO	0x0	reserved
8	RW	0x0	gpio8_srstn_req GPIO8 software reset request. When HIGH, reset relative logic
7	RW	0x0	gpio7_srstn_req GPIO7 software reset request. When HIGH, reset relative logic
6	RW	0x0	gpio6_srstn_req GPIO6 software reset request. When HIGH, reset relative logic

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	gpio5_srstn_req GPIO5 software reset request. When HIGH, reset relative logic
4	RW	0x0	gpio4_srstn_req GPIO4 software reset request. When HIGH, reset relative logic
3	RW	0x0	gpio3_srstn_req GPIO3 software reset request. When HIGH, reset relative logic
2	RW	0x0	gpio2_srstn_req GPIO2 software reset request. When HIGH, reset relative logic
1	RW	0x0	gpio1_srstn_req GPIO1 software reset request. When HIGH, reset relative logic
0	RW	0x0	gpio0_srstn_req GPIO0 software reset request. When HIGH, reset relative logic

**CRU\_SOFTRST3\_CON**

Address: Operational Base + offset (0x01c4)

Internal software reset control register3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	usb_peri_srstn_req USB PERIPH software reset request. When HIGH, reset relative logic
14	RW	0x0	emem_peri_srstn_req EMEM ahb bus software reset request. When HIGH, reset relative logic
13	RW	0x0	pd_peri_ahb_arbitor_srstn_req pd_peri ahb arbitor software reset request. cypro, nandc, hsic, otg, uhost AHB arbitor reset control When HIGH, reset relative logic
12	RW	0x0	periph_niu_srstn_req PERIPH NIU software reset request. When HIGH, reset relative logic

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x0	periphsys_psrstn_req PERIPH APB software reset request. pd_peri bus matrix apb softreset When HIGH, reset relative logic
10	RW	0x0	periphsys_hsrstn_req PERIPH AHB software reset request. pd_peri bus matrix ahb softreset When HIGH, reset relative logic
9	RW	0x0	periphsys_asrstn_req PERIPH AXI software reset request. pd_peri bus matrix axi softreset When HIGH, reset relative logic
8	RW	0x0	pmu_srstn_req PMU software reset request. When HIGH, reset relative logic
7	RW	0x0	grf_srstn_req GRF software reset request. When HIGH, reset relative logic
6	RW	0x0	pmu_psrstn_req PMU APB bus software reset request. When HIGH, reset relative logic
5	RW	0x0	tpiu_atrsrstn_req TPIU ATB software reset request. When HIGH, reset relative logic
4	RW	0x0	dap_sys_srstn_req DAP system software reset request. When HIGH, reset relative logic
3	RW	0x0	dap_srstn_req DAP software reset request. When HIGH, reset relative logic
2	RW	0x0	periph_mmu_srstn_req PERIPH MMU software reset request. When HIGH, reset relative logic
1	RW	0x0	mmc_peri_srstn_req pd_peri mmc AHB bus software reset request. emmc, sdio, sdmmc AHB arbitor reset control When HIGH, reset relative logic
0	RW	0x0	dw_pwm_srstn_req DW_PWM software reset request. When HIGH, reset relative logic

**CRU\_SOFTRST4\_CON**

Address: Operational Base + offset (0x01c8)

Internal software reset control register4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RO	0x0	reserved
14	RW	0x0	nandc1_srstn_req NANDC1 software reset request. When HIGH, reset relative logic
13	RW	0x0	nandc0_srstn_req NANDC0 software reset request. When HIGH, reset relative logic
12	RW	0x0	hsadc_srstn_req HSADC software reset request. When HIGH, reset relative logic
11	RW	0x0	hsicphy_srstn_req HSICPHY software reset request. When HIGH, reset relative logic
10	RW	0x0	hsic_aux_srstn_req HSIC AUX AHB software reset request. When HIGH, reset relative logic
9	RW	0x0	hsic_srstn_req HSIC AHB software reset request. When HIGH, reset relative logic
8	RW	0x0	usb_host0_srstn_req USB HOST0 AHB software reset request. When HIGH, reset relative logic
7	RW	0x0	ccp_srstn_req CCP software reset request. When HIGH, reset relative logic
6	RO	0x0	reserved
5	RW	0x0	rk_pwm_srstn_req RK_PWM software reset request. When HIGH, reset relative logic
4	RO	0x0	reserved
3	RW	0x0	gps_srstn_req GPS software reset request. When HIGH, reset relative logic
2	RW	0x0	mac_srstn_req MAC software reset request. When HIGH, reset relative logic
1	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	dma2_srstn_req DMA2 software reset request. When HIGH, reset relative logic

**CRU\_SOFTRST5\_CON**

Address: Operational Base + offset (0x01cc)

Internal software reset control register5

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:12	RO	0x0	reserved
11	RW	0x0	security_grf_psrstn_req security GRF APB software reset request. When HIGH, reset relative logic
10	RW	0x0	pd_pmu_niu_psrstn_req pd_pmu niu APB software reset request. When HIGH, reset relative logic
9	RW	0x0	pd_pmu_intmem_psrstn_req pd_pmu internal memory apb software reset request. When HIGH, reset relative logic
8	RW	0x0	pd_alive_niu_psrstn_req pd_alive niu APB software reset request. When HIGH, reset relative logic
7	RW	0x0	saradc_srstn_req SARADC software reset request. When HIGH, reset relative logic
6	RO	0x0	reserved
5	RW	0x0	spi2_srstn_req SPI2 software reset request. When HIGH, reset relative logic
4	RW	0x0	spi1_srstn_req SPI1 software reset request. When HIGH, reset relative logic
3	RW	0x0	spi0_srstn_req SPI0 software reset request. When HIGH, reset relative logic
2:1	RO	0x0	reserved
0	RW	0x0	tzpc_srstn_req TZPC software reset request. When HIGH, reset relative logic

**CRU\_SOFTRST6\_CON**

Address: Operational Base + offset (0x01d0)

Internal software reset control register6

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	edp_srstn_req eDP software reset request. When HIGH, reset relative logic
14	RW	0x0	isp_srstn_req ISP software reset request. When HIGH, reset relative logic
13	RW	0x0	rga_hsrstn_req RGA AHB software reset request. When HIGH, reset relative logic
12	RW	0x0	rga_asrstn_req RGA AXI software reset request. When HIGH, reset relative logic
11	RW	0x0	iep_hsrstn_req IEP AHB software reset request. When HIGH, reset relative logic
10	RW	0x0	iep_asrstn_req IEP AXI software reset request. When HIGH, reset relative logic
9	RW	0x0	rga_core_srstn_req RGA func software reset request. When HIGH, reset relative logic
8	RW	0x0	vip_srstn_req VIP software reset request. IEP ISP VOP's NIU software reset. When HIGH, reset relative logic
7	RW	0x0	vio1_niu_asrstn_req VIO1 NIU AXI software reset request. IEP ISP VOP's NIU software reset. When HIGH, reset relative logic
6	RW	0x0	lcdc0_dsrstn_req LCDC0 DCLK software reset request. When HIGH, reset relative logic
5	RW	0x0	lcdc0_hsrstn_req LCDC0 AHB software reset request. When HIGH, reset relative logic

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RW	0x0	lcdc0_asrstn_req LCDC0 AXI software reset request. When HIGH, reset relative logic
3	RW	0x0	vio_niu_hsrstn_req VIO NIU AHB software reset request. When HIGH, reset relative logic
2	RW	0x0	vio0_niu_asrstn_req VIO0 NIU AXI software reset request. IEP ISP VOP's NIU software reset. When HIGH, reset relative logic
1	RW	0x0	rga_niu_asrstn_req RGA NIU AXI software reset request. IEP ISP VOP's NIU software reset. When HIGH, reset relative logic
0	RW	0x0	vio_arbi_hsrstn_req VIO arbitor AHB software reset request. When HIGH, reset relative logic

**CRU\_SOFTRST7\_CON**

Address: Operational Base + offset (0x01d4)

Internal software reset control register7

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved
13	RW	0x0	gpu_pvtm_srstn_req gpu pvtm software reset request. When HIGH, reset relative logic
12	RW	0x0	core_pvtm_srstn_req core pvtm software reset request. When HIGH, reset relative logic
11:10	RO	0x0	reserved
9	RW	0x0	hdmi_srstn_req HDMI software reset request. When HIGH, reset relative logic
8	RW	0x0	gpu_srstn_req GPU core software reset request. When HIGH, reset relative logic
7	RW	0x0	lvds_con_srstn_req LVDS controller software reset request. When HIGH, reset relative logic

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RW	0x0	lvds_phy_psrstn_req LVDS PHY APB software reset request. When HIGH, reset relative logic
5	RW	0x0	mipicsi_psrstn_req MIPI CSI APB software reset request. When HIGH, reset relative logic
4	RW	0x0	mipidsi1_psrstn_req MIPI DSI1 APB software reset request. When HIGH, reset relative logic
3	RW	0x0	mipidsi0_psrstn_req MIPI DSI0 APB software reset request. When HIGH, reset relative logic
2	RW	0x0	vio_h2p_hsrstn_req VIO ahb to apb bridge AHB software reset request. When HIGH, reset relative logic
1	RW	0x0	vcodec_hsrstn_req VCODEC AHB software reset request. When HIGH, reset relative logic
0	RW	0x0	vcodec_asrstn_req VCODEC AXI software reset request. When HIGH, reset relative logic

**CRU\_SOFTRST8\_CON**

Address: Operational Base + offset (0x01d8)

Internal software reset control register8

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RO	0x0	reserved
14	RW	0x0	acc_efuse_srstn_req acc efuse software reset request. When HIGH, reset relative logic
13	RW	0x0	usb_adp_srstn_req OTG adp clock software reset request. When HIGH, reset relative logic
12	RW	0x0	usbhost1c_srstn_req USBHOST1 controller software reset request. When HIGH, reset relative logic

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x0	usbhost1phy_srstn_req USBHOST1 PHY software reset request. When HIGH, reset relative logic
10	RW	0x0	usbhost1_hsrstn_req USBHOST1 AHB BUS software reset request. When HIGH, reset relative logic
9	RW	0x0	usbhost0c_srstn_req USBHOST0 controller software reset request. When HIGH, reset relative logic
8	RW	0x0	usbhost0phy_srstn_req USBHOST0 PHY software reset request. When HIGH, reset relative logic
7	RW	0x0	usbhost0_hsrstn_req USBHOST0 AHB BUS software reset request. When HIGH, reset relative logic
6	RW	0x0	usbotgc_srstn_req USBOTG controller software reset request. When HIGH, reset relative logic
5	RW	0x0	usbotgphy_srstn_req USBOTG PHY software reset request. When HIGH, reset relative logic
4	RW	0x0	usbotg_hsrstn_req USBOTG AHB BUS software reset request. When HIGH, reset relative logic
3	RW	0x0	emmc_srstn_req EMMC software reset request. When HIGH, reset relative logic
2	RW	0x0	sdio1_srstn_req SDIO1 software reset request. When HIGH, reset relative logic
1	RW	0x0	sdio0_srstn_req SDIO0 software reset request. When HIGH, reset relative logic
0	RW	0x0	mmc0_srstn_req SDMMC0 software reset request. When HIGH, reset relative logic

**CRU\_SOFTRST9\_CON**

Address: Operational Base + offset (0x01dc)

Internal software reset control register9

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	tsadc_psrstn_req TSADC APB software reset request. When HIGH, reset relative logic
14:11	RO	0x0	reserved
10	RW	0x0	hevc_srstn_req HEVC software reset request. When HIGH, reset relative logic
9	RW	0x0	rga_h2p_brg_srstn_req RGA AHB to APB bridge software reset request. When HIGH, reset relative logic
8	RW	0x0	vio1_h2p_brg_srstn_req VIO1 AHB to APB bridge software reset request. When HIGH, reset relative logic
7	RW	0x0	vio0_h2p_brg_srstn_req VIO0 AHB to APB bridge software reset request. When HIGH, reset relative logic
6	RW	0x0	lcddcpwm1_srstn_req lcddc_pwm1 software reset request. When HIGH, reset relative logic
5	RW	0x0	lcddcpwm0_srstn_req lcddc_pwm0 software reset request. When HIGH, reset relative logic
4	RW	0x0	gic_srstn_req GIC software reset request. When HIGH, reset relative logic
3	RW	0x0	pd_core_mp_axi_srstn_req pd_croe periph axi software reset request. When HIGH, reset relative logic
2	RW	0x0	pd_core_apb_noc_srstn_req pd_core APB software reset request. When HIGH, reset relative logic
1	RW	0x0	pd_core_ahb_noc_srstn_req PD_CORE AHB software reset request. When HIGH, reset relative logic

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	coresight_srstn_req coresight software reset request. When HIGH, reset relative logic

**CRU\_SOFTRST10\_CON**

Address: Operational Base + offset (0x01e0)

Internal software reset control register10

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	c2c_host_srstn_req c2c host clk domain software reset request. When HIGH, reset relative logic
14	RW	0x0	crypto_srstn_req crypto working clk domain software reset request. When HIGH, reset relative logic
13:12	RO	0x0	reserved
11	RW	0x0	ddrmsch1_srstn_req DDR1 memory scheduler software reset request. When HIGH, reset relative logic
10	RW	0x0	ddrmsch0_srstn_req DDR0 memory scheduler software reset request. When HIGH, reset relative logic
9	RW	0x0	ddrphy1_ctl_srstn_req DDR1 PUB software reset request. When HIGH, reset relative logic
8	RW	0x0	ddrctrl1_psrstn_req DDR controller1 APB software reset request. When HIGH, reset relative logic
7	RW	0x0	ddrctrl1_srstn_req DDR controller1 software reset request. When HIGH, reset relative logic
6	RW	0x0	ddrphy1_psrstn_req DDR PHY1 APB software reset request. When HIGH, reset relative logic
5	RW	0x0	ddrphy1_srstn_req DDR PHY1 software reset request. When HIGH, reset relative logic

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RW	0x0	ddrphy0_ctl_srstn_req DDR0 PUB software reset request. When HIGH, reset relative logic
3	RW	0x0	ddrctrl0_psrstn_req DDR controller0 APB software reset request. When HIGH, reset relative logic
2	RW	0x0	ddrctrl0_srstn_req DDR controller0 software reset request. When HIGH, reset relative logic
1	RW	0x0	ddrphy0_psrstn_req DDR PHY0 APB software reset request. When HIGH, reset relative logic
0	RW	0x0	ddrphy0_srstn_req DDR PHY0 software reset request. When HIGH, reset relative logic

**CRU\_SOFTRST11\_CON**

Address: Operational Base + offset (0x01e4)

Internal software reset control register11

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	tsp_27m_srstn_req TSP 27M lock domain software reset request. When HIGH, reset relative logic
14	RW	0x0	tsp_clkin1_srstn_req TSP clockin1 software reset request. When HIGH, reset relative logic
13	RW	0x0	tsp_clkin0_srstn_req TSP clockin 0 software reset request. When HIGH, reset relative logic
12	RW	0x0	tsp_srstn_req tsp software reset request. When HIGH, reset relative logic
11	RW	0x0	ps2c_srstn_req ps2 controlor software reset request. When HIGH, reset relative logic
10	RW	0x0	simc_srstn_req cim card controlor software reset request. When HIGH, reset relative logic
9:8	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	uart4_srstn_req UART4 software reset request. When HIGH, reset relative logic
6	RW	0x0	uart3_srstn_req UART3 software reset request. When HIGH, reset relative logic
5	RW	0x0	uart2_srstn_req UART2 software reset request. When HIGH, reset relative logic
4	RW	0x0	uart1_srstn_req UART1 software reset request. When HIGH, reset relative logic
3	RW	0x0	uart0_srstn_req UART0 software reset request. When HIGH, reset relative logic
2	RW	0x0	lcdc1_dsrstn_req LCD1 DCLK software reset request. When HIGH, reset relative logic
1	RW	0x0	lcdc1_hsrstn_req LCD1 AHB software reset request. When HIGH, reset relative logic
0	RW	0x0	lcdc1_asrstn_req LCD1 AXI software reset request. When HIGH, reset relative logic

**CRU\_MISC\_CON**

Address: Operational Base + offset (0x01e8)

SCU control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:12	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:8	RW	0x0	testclk_sel Output clock selection for test 4'b0000: aclk_periph 4'b0001: clk_core 4'b0010: aclk_vio0 4'b0011: clk_ddrphy 4'b0100: aclk_vcodec 4'b0101: aclk_gpu 4'b0110: clk_rga_core 4'b0111: aclk_cpu 4'b1000: 24MHz 4'b1001: 27MHz 4'b1010: 32KHz 4'b1011: clk_wifi(16.368MHz) 4'b1100: dclk_lcdc0 4'b1101: dclk_lcdc1 4'b1110: clk_isp_jpeg 4'b1111: clk_isp
7:0	RO	0x0	reserved

**CRU\_GLB\_CNT\_TH**

Address: Operational Base + offset (0x01ec)

global reset wait counter threshold

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9:0	RW	0x064	glb_RST_CNT_TH Global soft reset counter threshold

**CRU\_GLB\_RST\_CON**

Address: Operational Base + offset (0x01f0)

global reset trigger select

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:2	RW	0x0	pmu_glb_srst_ctrl pmu reset by global soft reset select 2'b00: pmu reset by first global soft reset 2'b01: pmu reset by second global soft reset 2'b10: pmu not reset by any global soft reset
1	RW	0x0	wdt_glb_srst_ctrl watch_dog trigger global soft reset select 1'b0: watch_dog trigger second global reset 1'b1: watch_dog trigger first global reset

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	tsadc_glb_srst_ctrl TSADC trigger global soft reset select 1'b0: tsadc trigger second global reset 1'b1: tsadc trigger first global reset

**CRU\_GLB\_RST\_ST**

Address: Operational Base + offset (0x01f8)

global reset status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	W1C	0x0	snd_glb_wdt_RST_st second global watch_dog triggered reset flag 1'b0: last hot reset is not second global watch_dog triggered reset 1'b1: last hot reset is second global watch_dog triggered reset
4	W1C	0x0	fst_glb_wdt_RST_st first global watch_dog triggered reset flag 1'b0: last hot reset is not first global watch_dog triggered reset 1'b1: last hot reset is first global watch_dog triggered reset
3	W1C	0x0	snd_glb_tsadc_RST_st second global TSADC triggered reset flag 1'b0: last hot reset is not second global TSADC triggered reset 1'b1: last hot reset is second global TSADC triggered reset
2	W1C	0x0	fst_glb_tsadc_RST_st first global TSADC triggered reset flag 1'b0: last hot reset is not first global TSADC triggered reset 1'b1: last hot reset is first global TSADC triggered reset
1	W1C	0x0	snd_glb_RST_st second global rst flag 1'b0: last hot reset is not second global rst 1'b1: last hot reset is second global rst
0	W1C	0x0	fst_glb_RST_st first global rst flag 1'b0: last hot reset is not first global rst 1'b1: last hot reset is first global rst

**CRU\_SDMMC\_CON0\***

Address: Operational Base + offset (0x0200)

## sdmmc control0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:12	RO	0x0	reserved
11	WO	0x0	sdmmc_drv_sel sdmmc drive select sdmmc drive select
10:3	WO	0x00	sdmmc_drv_delaynum sdmmc drive delay number sdmmc drive delay number
2:1	WO	0x1	sdmmc_drv_degree sdmmc drive degree sdmmc drive degree
0	WO	0x0	sdmmc_init_state sdmmc initial state sdmmc initial state

**CRU\_SDMMC\_CON1\***

Address: Operational Base + offset (0x0204)

## sdmmc control1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:11	RO	0x0	reserved
10	WO	0x0	sdmmc_sample_sel sdmmc sample select sdmmc sample select
9:2	WO	0x00	sdmmc_sample_delaynum sdmmc sample delay number sdmmc sample delay number
1:0	WO	0x0	sdmmc_sample_degree sdmmc sample degree sdmmc sample degree

**CRU\_SDIO0\_CON0\***Address: Operational Base + offset (0x0208)  
sdio0 control0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:12	RO	0x0	reserved
11	WO	0x0	sdio0_drv_sel sdio0 drive select sdio0 drive select
10:3	WO	0x00	sdio0_drv_delaynum sdio0 drive delay number sdio0 drive delay number
2:1	WO	0x1	sdio0_drv_degree sdio0 drive degree sdio0 drive degree
0	WO	0x0	sdio0_init_state sdio0 initial state sdio0 initial state

**CRU\_SDIO0\_CON1\***

Address: Operational Base + offset (0x020c)

sdio0 control1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:11	RO	0x0	reserved
10	WO	0x0	sdio0_sample_sel sdio0 sample select sdio0 sample select
9:2	WO	0x00	sdio0_sample_delaynum sdio0 sample delay number sdio0 sample delay number
1:0	WO	0x0	sdio0_sample_degree sdio0 sample degree sdio0 sample degree

**CRU\_SDIO1\_CON0\***

Address: Operational Base + offset (0x0210)

sdio1 control0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:12	RO	0x0	reserved
11	WO	0x0	sdio1_drv_sel sdio1 drive select sdio1 drive select
10:3	WO	0x00	sdio1_drv_delaynum sdio1 drive delay number sdio1 drive delay number
2:1	WO	0x1	sdio1_drv_degree sdio1 drive degree sdio1 drive degree
0	WO	0x0	sdio1_init_state sdio1 initial state sdio1 initial state

**CRU\_SDIO1\_CON1\***

Address: Operational Base + offset (0x0214)

sdio1 control1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:11	RO	0x0	reserved
10	WO	0x0	sdio1_sample_sel sdio1 sample select sdio1 sample select
9:2	WO	0x00	sdio1_sample_delaynum sdio1 sample delay number sdio1 sample delay number
1:0	WO	0x0	sdio1_sample_degree sdio1 sample degree sdio1 sample degree

**CRU\_EMMC\_CON0\***

Address: Operational Base + offset (0x0218)

emmc control0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:12	RO	0x0	reserved
11	WO	0x0	emmc_drv_sel emmc drive select emmc drive select
10:3	WO	0x00	emmc_drv_delaynum emmc drive delay number emmc drive delay number
2:1	WO	0x1	emmc_drv_degree emmc drive degree emmc drive degree
0	WO	0x0	emmc_init_state emmc initial state emmc initial state

**CRU\_EMMC\_CON1\***

Address: Operational Base + offset (0x021c)  
emmc control1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:11	RO	0x0	reserved
10	WO	0x0	emmc_sample_sel emmc sample select emmc sample select
9:2	WO	0x00	emmc_sample_delaynum emmc sample delay number emmc sample delay number
1:0	WO	0x0	emmc_sample_degree emmc sample degree emmc sample degree

\*Notes: CRU\_SDMMC\_CON0/1, CRU\_SDIO1\_CON0/1, CRU\_SDIO0\_CON0/1, CRU\_EMMC\_CON0/1, detail description please refer to chapter15 Mobile Storage Host Controller 15.6.10.

## 3.8 Timing Diagram

Power on reset timing is shown as follow:

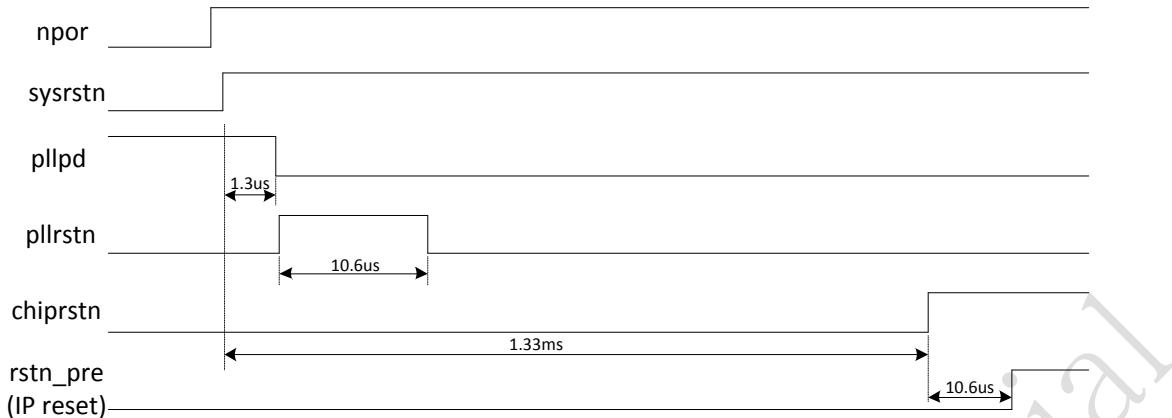


Fig. 3-8 Chip Power On Reset Timing Diagram

Npor is hardware reset signal from out-chip and power-off mode wakeup reset from PMU, which is filtered glitch to obtain signal sysrstn. To make PLLs work normally, the power down signal (pllpd) must be high when reset, and maintains high for more than 1us when sysrstn de-active. Then PLL reset signals (pllrstn) are asserted for about 10.6us, and PLLs start to lock when pllrstn de-assert, and consume about 1330us to lock. So the system will wait about 1330us, then de-active reset signal chiprstn. The signal chiprstn is used to generate output clocks in CRU. After CRU start output clocks, the system waits again for 256 cycles (10.7us) to de-active signal rstn\_pre, which is used to generate power on reset of all IP.

## 3.9 Application Notes

### 3.9.1 PLL usage

#### A. PLL output frequency configuration

The output frequency Fout is related to the input frequency Fin by:

$$F_{out} = ((F_{in} / NR) * NF) / NO$$

Fout is clock output of PLL, and Fin is clock input of PLL from external oscillators (24MHz). Another, other factors such as NF, NR, NO can be configured by programming CRU\_APLL\_CON*i*, CRU\_DPLL\_CON*i*, CRU\_CPLL\_CON*i* and CRU\_GPLL\_CON*i* registers (*i*=0,1,2), and their value will affect Fout as follows.

- (1) CLKR: A 6-bit bus that selects the values 1-64 for the reference divider (NR)

$$NR = CLKR[5:0] + 1$$

Example:

/1	pgm	000000
/4	pgm	000011
/8	pgm	000111

- (2) CLKF: A 13-bit bus that selects the values 1-4096 for the PLL multiplication factor (NF)

$$NF = CLKF[12:0] + 1$$

Example:

X1	pgm	0000000000000000
X2	pgm	0000000000000001

X4096 pgm 011111111111

- (3) CLKOD: A 4-bit bus that selects the value 1,2-16 (even only) for the PLL post VCO divider (NO)

$$NO = CLKOD[3:0] + 1$$

Example:

/1	pgm 0000
/2	pgm 0001
/4	pgm 0011
/8	pgm 0111

- (4) BWADJ: A 12-bit bus that selects the values 1-4096 for the bandwidth divider (NB)

$$NB = BWADJ[11:0] + 1$$

Example:

/1	pgm 000000000000
/4	pgm 000000000001
/8	pgm 000000000011

The recommended setting of NB:  $NB = NF / 2$ .

## B. PLL frequency range requirement

If different CLKR, CLKF and CLKOD configuration value cause internal out of range, unpredicted result will be caused.

Fin value range requirement:

269kHz – 2200MHz

Fref = Fin/NR value range requirement:

269kHz – 2200MHz

Fvco = (Fin/NR)\*NF value range requirement:

440MHz – 2200MHz

Fout = ((Fin/NR)\*NF)/NO value range requirement:

27.5MHz – 2200MHz

## C. PLL setting consideration

Optimization of the PLL settings for jitter < +/- 2.5% of the output period/sq rt(NO) require running the VCO at maximum frequency and dividing down using the NO divider to get the required Fout, i.e. maximum NO.

Optimization for minimum power ( $Fvco/1100MHz * 3.3 \text{ mA}$ ) requires setting the VCO frequency at the minimum frequency and using the lowest NO setting.

These two values, minimum jitter or minimum power will determine your choice of settings.

A larger value of input divider NR gives a longer lock time, and higher long term as well as period jitter. It is better to use a lower value of NR where possible.

### 3.9.2 PLL frequency change method

When the PLL settings are changed, it has to reset PLL by programming registers CRU\_APLL\_CON3, CRU\_DPLL\_CON3, CRU\_CPLL\_CON3, CRU\_GPLL\_CON3, CRU\_NPLL\_CON3, and reserve at least 5us after valid settings, referring to the following figure.

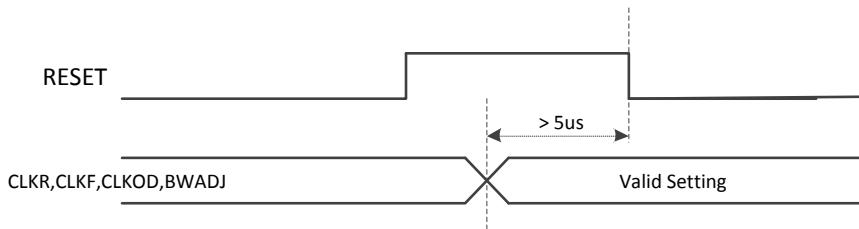


Fig. 3-9 PLL setting change timing

Before set some factors such as NR/NF/NO/BS to change PLL output frequency, you must change chip from normal to slow mode by programming CRU\_MODE\_CON. Then until PLL is lock state by checking GRF\_SOC\_STATUS0[8:5] register, or after delay about  $(NR * 500) / Fin$ , you can change PLL into normal mode.

### 3.9.3 Fractional divider usage

To get specific frequency, clocks of I2S, SPDIF, UART, HSADC can be generated by fractional divider. Generally you must set that denominator is 20 times larger than numerator to generate precise clock frequency. So the fractional divider applies only to generate low frequency clock like I2S, UART and HSADC.

All the fractional divider has auto-gating control. When fractional divider is not selected, the divider clock is gated. So fractional divider must be selected before changing configuration.

### 3.9.4 Global software reset

Two global software resets are designed in this chip, you can program CRU\_GLB\_SRST\_FST\_VALUE[15:0] as 0xfdb9 to assert the first global software reset glb\_srstn\_1 and program CRU\_GLB\_SRST\_SND\_VALUE[15:0] as 0xeca8 to assert the second global software reset glb\_srstn\_2. These two software resets are self-deasserted by hardware.

TSADC, WDT and PMU also can trigger glb\_srstn\_1 or glb\_srstn\_2.

CRU\_GLB\_RST\_CON controls which global soft-reset will be triggered.

After global reset, the reset trigger source can be check in CRU\_GLB\_RST\_ST.

glb\_srstn\_1 resets almost all chip logic except PMU\_SYS\_REG0~3, which can be used to store something when reset.

glb\_srstn\_2 resets almost all chip logic except PMU\_SYS\_REG0~3, GRF and GPIOs.

### 3.9.5 Pre-shift for test

Pre-shift registers is designed in this chip for flexible test.

The key configuration registers can be shifted with a initial value in testmode. The pre-shift registers including 191 bit pre\_shift\_test\_reg.

The following table describes the pre-shift registers of the design.

Name	Bit number	Default value	Description
armpll_clkr	pre_shift_test_reg[5:0]	6'd0	armpll_clkr control

armpll_clkf	pre_shift_test_reg[18:6]	13'd199	armpll_clkf control
armpll_bwadj	pre_shift_test_reg[30:19]	12'd49	armpll_bwadj control
armpll_clkod	pre_shift_test_reg[35:31]	5'd1	armpll_clkod control
ddrpll_clkr	pre_shift_test_reg[41:36]	6'd1	ddrpll_clkr control
ddrpll_clkf	pre_shift_test_reg[54:42]	13'd99	ddrpll_clkf control
ddrpll_bwadj	pre_shift_test_reg[66:55]	12'd49	ddrpll_bwadj control
ddrpll_clkod	pre_shift_test_reg[71:67]	5'd5	ddrpll_clkod control
codecpll_clkr	pre_shift_test_reg[77:72]	6'd1	codecpll_clkr control
codecpll_clkf	pre_shift_test_reg[90:78]	13'd99	codecpll_clkf control
codecpll_bwadj	pre_shift_test_reg[102:91]	12'd49	codecpll_bwadj control
codecpll_clkod	pre_shift_test_reg[107:103]	5'd5	codecpll_clkod control
generalpll_clkr	pre_shift_test_reg[113:108]	6'd1	generalpll_clkr control
generalpll_clkf	pre_shift_test_reg[126:114]	13'd99	generalpll_clkf control
generalpll_bwadj	pre_shift_test_reg[138:127]	12'd49	generalpll_bwadj control
generalpll_clkod	pre_shift_test_reg[143:139]	5'd5	generalpll_clkod control
newpll_clkr	pre_shift_test_reg[149:144]	6'd1	newpll_clkr control
newpll_clkf	pre_shift_test_reg[162:150]	13'd99	newpll_clkf control
newpll_bwadj	pre_shift_test_reg[174:163]	12'd49	newpll_bwadj control
newpll_clkod	pre_shift_test_reg[179:175]	5'd5	newpll_clkod control
testclk_sel	pre_shift_test_reg[182:180]	3'd0	testclk_out select in testmode
aclk_core_m_div_con	pre_shift_test_reg[185:183]	3'd1	Aclk_m divider configuration in testmode
Io_sr	pre_shift_test_reg[186]	1'd1	IO slew rate
io_drive	pre_shift_test_reg[188:187]	2'b10	IO drive configuration
Io_vsel	pre_shift_test_reg[189]	1'd0	IO voltage select
Io_smt	pre_shift_test_reg[190]	1'd0	IO smt control

Pre-shift relative controls IO are as follow.

Name	IO	description
Pre_shift_datain	IO_UART3GPSout_GPSsig_H SADCT1data1_GPIO30gpio7b0	Pre-shift data in
Pre_shift_en	IO_UART3GPSctsn_GPSrfclk_GPST1clk_GPIO30gpio7b1	Pre-shift enable
Pre_shift_clk	IO_UART3GPSrtsn_USBdrvbus0_GPIO30gpio7b2	Pre-shift clock
Pre-shift_default_select	IO_USBdrvbus1_EDPhotplug_GPIO30gpio7b3	1'b0: pre_shift use default value; 1'b1: pre_shift use shift in value;

Pre-shift_select	IO_ISPshutteren_SPI1clk_GP IO30gpio7b4	1'b0: disable, testmode do not use pre-shift config value; use internal 6 configs. 1'b1: enable, testmode use pre-shift config value;
------------------	---	--

## Chapter 4 Power Management Unit (PMU)

### 4.1 Overview

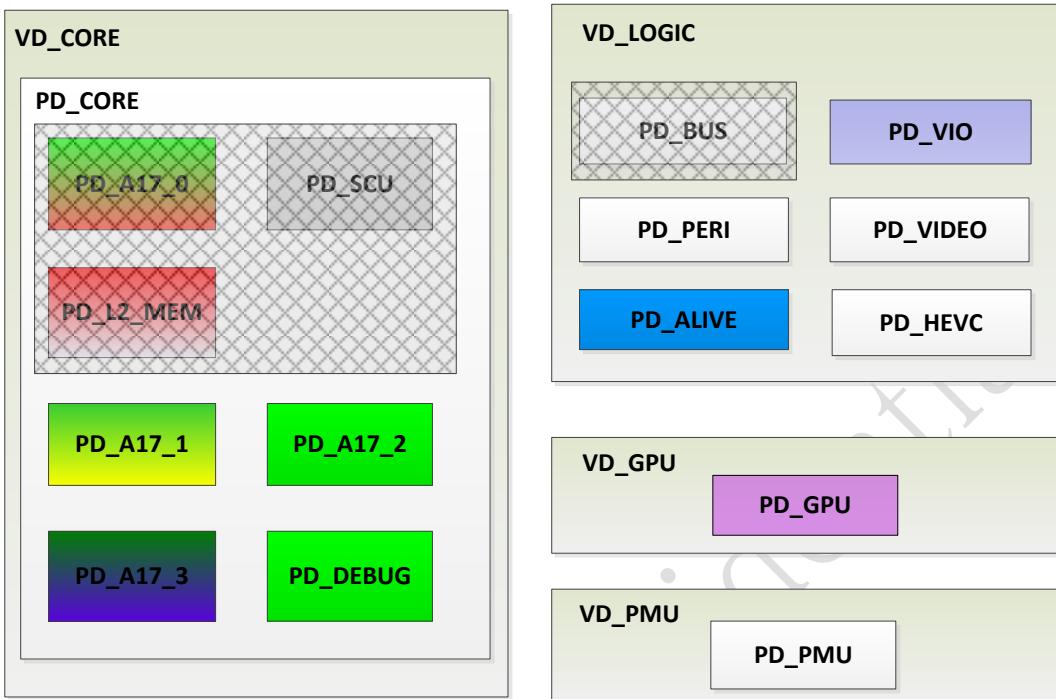
In order to meet high performance and low power requirements, a power management unit is designed for saving power when RK3288 in low power mode. The RK3288 PMU is dedicated for managing the power of the whole chip.

#### 4.1.1 Features

- Support 4 voltage domains including VD\_CORE, VD\_LOGIC, VD\_GPU and VD\_PMU
- Support 15 separate power domains in the whole chip, which can be power up/down by software based on different application scenes
- In low power mode, the pmu could power up/down pd\_A17\_0, pd\_scu, vd\_core, and pd\_bus by hardware
- Support CORTEX-A17 core source clock gating in low power mode
- Support global interrupt disable in low power mode
- Support PLLs power down/up in low power mode
- Support VD\_CORE/VD\_LOGIC power down/up in low power mode
- Support pd\_alive clock switch to 32KHz in low power mode
- Support pd\_pmu clock switch to 32KHz request in low power mode
- Support OSC enable/disable request in low power mode
- Support to clamp all VD\_LOGIC output before power off it in low power mode
- Support wakeup reset control in power off mode
- Support DDR self-refresh in low power mode
- Support DDR IO retention in low power mode
- Support DDR IO power off in low power mode
- Support DDR controller clock auto gating in low power mode
- Support to send idle requests to all NIU in the SoC (details will be described later)
- A group of configurable counter in PMU for HW control (such as PLL, PMIC, DDRIIO and so on)
- Support varies configurable wakeup source for low power mode

## 4.2 Block Diagram

### 4.2.1 power domain partition



Note:

VD\_\* : voltage domain

PD\_\* : power domain

Fig. 4-1 Power Domain Partition

The above diagram describes the power domain and voltage domain partition, and the following table lists all the power domains.

Table 4-1 RK3288 Power Domain and Voltage Domain Summary

Voltage Domain	Power Domain	Description
VD_CORE (PD_CORE system)	PD_A17_0	A17 primary core logic, L1C and noen
	PD_A17_1	A17 slave core 1 logic, L1C and noen
	PD_A17_2	A17 slave core 2 logic, L1C and noen
	PD_A17_3	A17 slave core 3 logic, L1C and noen
	PD_SCU	SCU RAM, SCU, GIC, Periphral, L2 controller
	PD_DEBUG	A17 Debug
	PD_MEM	L2 Cache
VD_LOGIC	PD_BUS	Soc architecture subsystem, include soc architecture (NOC) , eFuse, TZPC, ROM, DMAC_BUS, Crypto, Host, Timer(6ch), PWM(0~3), UART_DBG, I2C, DDR_PCTL, I2S, Spdif, Internal Memory(96K)
	PD_PERI	Peripheral subsystem , include DMAC_PERI, GMAC, NANDC0/1, HSIC/USB Host0/USB Host1/ USB OTG, SDMMC/SDIO0/SDIO1/eMMC, HSADC, PS2C, TSADC, UART, I2C, SPI, GPS, TSP
	PD_VIO	Video input/output system, include VOPBIG,

		VOPLIT, ISP, IEP, RGA, MIPI-CSI, MIPI-DSI, LVDS, HDMI, eDP
	PD_ALIVE	CRU, GRF, GPIO 1~8, TIMER, WDT
	PD_HEVC	HEVC
	PD_VIDEO	Video Encode&Decode , include VEPU, VDPU
VD_GPU	PD_GPU	GPU
VD_PMU	PD_PMU	PMU, SRAM(4K), Secure GRF, GPIO0

Notes: "Always on" means that their power supply can be switched off only by external PMIC module. Only one "always on" power domain is in a voltage domain.

#### 4.2.2 PMU block diagram

The following figure is the PMU block diagram. The PMU includes the 3 following sections:

- APB interface and register, which can accept the system configuration
- Low Power State Control, which generate low power control signals.
- Power Switch Control, which control all power domain switch

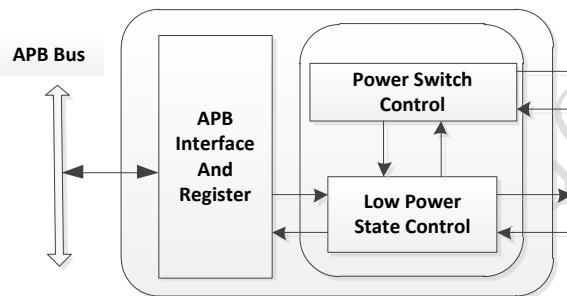


Fig. 4-2 PMU Bock Diagram

#### 4.3 Power Switch Timing Requirement

The following table describe the switch time for power down and power up progress of each power domain. This table gives the time range, and each power domain switch time will be more than the min time and less than the max time.

Table 4-2 Power Switch Timing

Power domain	type	Power down Switch Timing① (ns)	Power up Switch Timing① (ns)
PD_A17_0	min	170.3	132.4
	max	306.7	237.5
PD_A17_1	min	170.3	132.4
	max	306.7	237.5
PD_A17_2	min	181.5	140.7
	max	326.2	251.9
PD_A17_3	min	181.5	140.7
	max	326.2	251.9
PD_DEBUG	min	169.4	131.7
	max	77.3	60
PD_BUS	min	169.4	131.7
	max	313.6	247
PD_PERI	min	103.7	80.5
	max	199.0	156.1

PD_VIO	min	280.6	217.5
	max	518.5	407.8
PD_VIDEO	min	315.4	244.2
	max	586.2	460.4
PD_GPU	min	470.2	364
	max	871.4	684.1
PD_HEVC	min	33.4	25.9
	max	62.4	49
	max	65.6	51.5

*Notes: the power switch timing is just the chip power electrical parameter, this is not the parameter for the software to determine the power domain status. The software need to check each power domain status register to determine the power status.*

## 4.4 Function Description

### 4.4.1 Normal Mode

First of all, we define two modes of power for chip, normal mode and low power mode.

In normal mode, the PMU can power off/on all power domain (except pd\_A17\_0 and pd\_bus) by setting PMU\_PWRDN\_CON register. At same, pmu can send idle request for every power domain by setting PMU\_IDLE\_REQ register.

Don't set pd\_A17\_0 and pd\_bus power off or send idle\_req\_core and idle\_req\_bus in normal mode. This will cause the system to not work properly.

Basically, there are 2 configurations that software can do in normal mode to save power.

- Configure DDR to self-refresh, DDR IO retention and DDR IO power off
- Power down power domains

The first one will save power consumption of using DDR controller and DDR IO. For avoiding confliction, the software must make sure the execution code of this step is not in DDR.

The second one will save power of the power domain which software is shutting down.

### 4.4.2 Low Power Mode

PMU can work in the Low Power Mode by setting bit[0] of PMU\_PWRMODE\_CON register. After setting the register, PMU would enter the Low Power mode. In the low power mode, pmu will auto power on/off the specified power domain, send idle req to specified power domain, shut down/up pll and so on. All of above are configurable by setting corresponding registers.

Table 4-3 Low Power State

Num	Hardware Flow	Description of Flow	Corresponding Register
0	NORMAL	in normal	
1	L2FLUSH_REQ	send L2 cache flush request	bit[3] of PMU_PWRMODE_CON
2	STANDBYL2	wait L2 cache standy	
3	A17_CLK_DIS	close A17 clock	bit[1] of PMU_PWRMODE_CON
4	TRANS_NO_FIN	wait the corresponding noc interface end the transaction	PMU_PWRMODE_CON1
5	SREF_ENTER	enter DDR self-refresh	bit[16:15] of PMU_PWRMODE_CON
6	DDR_IO_RET	ddr io retention	bit[18:17] of PMU_PWRMODE_CON
7	DDR_IO_PWRROFF	ddr io power off	bit[20:19] of PMU_PWRMODE_CON
8	BUS_PWRDN	pd_bus power down	bit[4] of PMU_PWRMODE_CON
9	A17_0_PWRDN	pd_a17_0 power down	bit[5] of PMU_PWRMODE_CON
10	L2MEM_PWRDN	pd_l2mem power down (vd_core power down)	bit[6] of PMU_PWRMODE_CON
11	ALIVE_PMU_LF	pd_alive& pd_pmu switch to 32KHz clock	bit[11:10] of PMU_PWRMODE_CON
12	PLL_PWRDN	pll power down	bit[7] of PMU_PWRMODE_CON
13	INPUT_CLAMP	isolation cell input clamp	bit[13] of PMU_PWRMODE_CON
14	POWEROFF	chip power off	bit[8] of PMU_PWRMODE_CON
15	24M_OSC_DIS	close 24MHz OSC	bit[12] of PMU_PWRMODE_CON
16	WAIT_WAKEUP	wait wakeup source	PMU_WAKEUP_CFG0/1
17	WAKEUP_RESET	send reset after wakeup	
18	EXT_PWRUP	pmic power up whole chip	
19	RELEASE_CLAMP	release isolation cell clamp	
20	24M_OSC_EN	open 24MHz OSC	
21	ALIVE_PMU_HF	switch pd_alive and pd_pmu back to 24MHz	
22	WAKEUP_RESET_CLR	release wakeup reset	
23	PLL_PWRUP	PLL power up	
24	BUS_PWRUP	PD_BUS power up	
25	DDR_IO_PWRUP	ddr io power up	
26	SREF_EXIT	exit ddr self-refresh	
27	L2MEM_PWRUP	pd_l2mem power up	
28	A17_0_PWRUP	pd_a17_0 power up	
29	TRANS_RESTORE	restore the transaction	
30	A17_CLK_EN	enable a17 clock	

The Low Power mode have three steps:

- Enter Low Power mode, there are some sub-steps in the enter step, every sub-step can be enable/disable by setting the corresponding register.
- Wait wakeup, you can select the wakeup source by setting PMU\_WAKEUP\_CFG0/1 register
- Exit Low Power mode, the sub-step are executed depend on whether they were executed in enter low power step.

#### 4.4.3 Wakeup source of AP

The wakeup source is a group of signals which can trigger PMU from power mode to normal mode, such as SDMMC detect, core interrupt, GPIO0, and gpio interrupt.

Table 4-4 Wakeup Source

Num	Wakeup Source	Description
1	software control	software control (in normal mode)
2	arm interrupt	a17 interrupt
3	pmu_gpio	gpio0, in pd_pmu
4	sdmmc0	detect_n of sdmmc0
5	gpio int	gpio interrupt outside of pd_pmu

If software expect PMU be woken up from power mode by a wake up source, it should be enabled by write 1 to the corresponding bit of PMU\_WAKEUP\_CFG0/1 register before entering into power mode.

Technically, wakeup source can be used in every power mode if only the path from wakeup source to PMU is not shut down.

## 4.5 Register Description

### 4.5.1 Register Summary

Name	Offset	Size	Reset Value	Description
PMU_WAKEUP_CFG0	0x0000	W	0x00000000	PMU wake-up source configuration register0
PMU_WAKEUP_CFG1	0x0004	W	0x00000000	PMU wake-up source configuration register1
PMU_PWRDN_CON	0x0008	W	0x00000000	System power gating configuration register
PMU_PWRDN_ST	0x000c	W	0x00000000	System power gating status register
PMU_IDLE_REQ	0x0010	W	0x00000000	PMU Noc idle req control
PMU_IDLE_ST	0x0014	W	0x00000000	PMU Noc idle status
PMU_PWRMODE_CO_N	0x0018	W	0x00000000	PMU configuration register in power mode flow
PMU_PWR_STATE	0x001c	W	0x00000000	PMU Low power mode state
PMU_OSC_CNT	0x0020	W	0x00005dc0	24MHz OSC stabilization counter threshold
PMU_PLL_CNT	0x0024	W	0x10004000	PLL lock counter threshold
PMU_STABL_CNT	0x0028	W	0x00005dc0	External PMU stabilization counter threshold
PMU_DDR0IO_PWRO_N_CNT	0x002c	W	0x00005dc0	DDR0 IO power on counter threshold
PMU_DDR1IO_PWRO_N_CNT	0x0030	W	0x00005dc0	DDR1 IO power on counter threshold
PMU_CORE_PWRDW_N_CNT	0x0034	W	0x00005dc0	CORE domain power down waiting counter in sleep mode
PMU_CORE_PWRUP_CNT	0x0038	W	0x00005dc0	CORE domain power up waiting counter in sleep mode
PMU_GPU_PWRDWN_CNT	0x003c	W	0x00005dc0	GPU domain power down waiting counter in sleep mode
PMU_GPU_PWRUP_CNT	0x0040	W	0x00005dc0	GPU domain power up waiting counter in sleep mode
PMU_WAKEUP_RST_CLR_CNT	0x0044	W	0x00005dc0	Wakeup reset deassert state wait counter in power off mode
PMU_SFT_CON	0x0048	W	0x00000000	PMU Software control in normal mode
PMU_DDR_SREF_ST	0x004c	W	0x00000000	PMU DDR self refresh status
PMU_INT_CON	0x0050	W	0x00000000	PMU interrupt configuration register
PMU_INT_ST	0x0054	W	0x00000000	PMU interrupt status register

Name	Offset	Size	Reset Value	Description
PMU_BOOT_ADDR_SEL	0x0058	W	0x00005dc0	boot_addr_sel in power mode
PMU_GRF_CON	0x005c	W	0x00000008	grf control register
PMU_GPIO_SR	0x0060	W	0x00020000	GPIO slew rate control
PMU_GPIO0_A_PULL	0x0064	W	0x0000555a	GPIO0A input to PU/PD programmation section
PMU_GPIO0_B_PULL	0x0068	W	0x00005555	GPIO0B input to PU/PD programmation section
PMU_GPIO0_C_PULL	0x006c	W	0x00000015	GPIO0C input to PU/PD programmation section
PMU_GPIO0_A_DRV	0x0070	W	0x0000555a	GPIO0A Drive strength slector
PMU_GPIO0_B_DRV	0x0074	W	0x00005555	GPIO0B Drive strength slector
PMU_GPIO0_C_DRV	0x0078	W	0x00000015	GPIO0C Drive strength slector
PMU_GPIO_OP	0x007c	W	0x00000000	GPIO0 output value
PMU_GPIO0_SEL18	0x0080	W	0x00000006	gpio0 1.8v/3.3v sel
PMU_GPIO0_A_IOMUX	0x0084	W	0x00000000	GPIO0A iomux sel
PMU_GPIO0_B_IOMUX	0x0088	W	0x00000000	GPIO0B iomux sel
PMU_GPIO0_C_IOMUX	0x008c	W	0x00000000	GPIO0C iomux sel
PMU_PWRMODE_CO_N1	0x0090	W	0x00000000	PMU power mode controll1
PMU_SYS_REG0	0x0094	W	0x00000000	PMU system register0
PMU_SYS_REG1	0x0098	W	0x00000000	PMU system register1
PMU_SYS_REG2	0x009c	W	0x00000000	PMU system register2
PMU_SYS_REG3	0x00a0	W	0x00000000	PMU system register3

#### 4.5.2 Detail Register Description

##### PMU\_WAKEUP\_CFG0

Address: Operational Base + offset (0x0000)

PMU wake-up source configuration register0

Bit	Attr	Reset Value	Description
31:19	RO	0x0	reserved
18	RW	0x0	gpio0c_2_wakeup_en GPIO0c bit2 wakeup enable 1'b0: disable 1'b1: enable
17	RW	0x0	gpio0c_1_wakeup_en GPIO0c bit1 wakeup enable 1'b0: disable 1'b1: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16	RW	0x0	gpio0c_0_wakeup_en GPIO0c bit0 wakeup enable 1'b0: disable 1'b1: enable
15	RW	0x0	gpio0b_7_wakeup_en GPIO0b bit7 wakeup enable 1'b0: disable 1'b1: enable
14	RW	0x0	gpio0b_6_wakeup_en GPIO0b bit6 wakeup enable 1'b0: disable 1'b1: enable
13	RW	0x0	gpio0b_5_wakeup_en GPIO0b bit5 wakeup enable 1'b0: disable 1'b1: enable
12	RW	0x0	gpio0b_4_wakeup_en GPIO0b bit12 wakeup enable 1'b0: disable 1'b1: enable
11	RW	0x0	gpio0b_3_wakeup_en GPIO0b bit3 wakeup enable 1'b0: disable 1'b1: enable
10	RW	0x0	gpio0b_2_wakeup_en GPIO0b bit2 wakeup enable 1'b0: disable 1'b1: enable
9	RW	0x0	gpio0b_1_wakeup_en GPIO0b bit1 wakeup enable 1'b0: disable 1'b1: enable
8	RW	0x0	gpio0b_0_wakeup_en GPIO0b bit0 wakeup enable 1'b0: disable 1'b1: enable
7	RW	0x0	gpio0a_7_wakeup_en GPIO0a bit7 wakeup enable 1'b0: disable 1'b1: enable
6	RW	0x0	gpio0a_6_wakeup_en GPIO0a bit6 wakeup enable 1'b0: disable 1'b1: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	gpio0a_5_wakeup_en GPIO0a bit5 wakeup enable 1'b0: disable 1'b1: enable
4	RW	0x0	gpio0a_4_wakeup_en GPIO0a bit4 wakeup enable 1'b0: disable 1'b1: enable
3	RW	0x0	gpio0a_3_wakeup_en GPIO0a bit3 wakeup enable 1'b0: disable 1'b1: enable
2	RW	0x0	gpio0a_2_wakeup_en GPIO0a bit2 wakeup enable 1'b0: disable 1'b1: enable
1	RW	0x0	gpio0a_1_wakeup_en GPIO0a bit1 wakeup enable 1'b0: disable 1'b1: enable
0	RW	0x0	gpio0a_0_wakeup_en GPIO0a bit0 wakeup enable 1'b0: disable 1'b1: enable

**PMU\_WAKEUP\_CFG1**

Address: Operational Base + offset (0x0004)

PMU wake-up source configuration register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3	RW	0x0	gpioint_wakeup_en GPIO Interrupt wake enable 1'b0: disable 1'b1: enable
2	RW	0x0	sdmmc0_wakeup_en SDMMC0 wake-up enable 1'b0: disable 1'b1: enable
1	RW	0x0	pmu_gpio_wakeup_type GPIO in pmu wakeup type 1'b0: posedge 1'b1: negedge
0	RW	0x0	armint_wakeup_en ARM interrupt wake-up enable 1'b0: disable 1'b1: enable

**PMU\_PWRDN\_CON**

Address: Operational Base + offset (0x0008)

System power gating configuration register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	RO	0x0	reserved
14	RW	0x0	PD_HEVC_DWN_EN Power domain HEVC power down enable 1'b0: power on 1'b1: power off
13	RW	0x0	CHIP_PWROFF_EN software config power off chip logic 1'b1: power off 1'b0: not power off
12	RW	0x0	CORE_PWROFF_EN software config power off pd_core 1'b1: power off 1'b0: not power off
11	RW	0x0	PD_SCU_DWN_EN Power domain SCU power down enable 1'b0: power on 1'b1: power off
10	RO	0x0	reserved
9	RW	0x0	PD_GPU_DWN_EN Power domain GPU power down enable 1'b0: power on 1'b1: power off
8	RW	0x0	PD_VIDEO_DWN_EN Power domain VIDEO power down enable 1'b0: power on 1'b1: power off
7	RW	0x0	PD_VIO_DWN_EN Power domain VIO power down enable 1'b0: power on 1'b1: power off
6	RW	0x0	PD_PERI_DWN_EN Power domain PERI power down enable 1'b0: power on 1'b1: power off
5	RW	0x0	PD_BUS_DWN_EN Power domain BUS power down enable 1'b0: power on 1'b1: power off
4	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	PD_A17_3_DWN_EN Power Domain A17 slave core 3 power down enable 1'b0: power on 1'b1: power off
2	RW	0x0	PD_A17_2_DWN_EN Power Domain A17 slave core 2 power down enable 1'b0: power on 1'b1: power off
1	RW	0x0	PD_A17_1_DWN_EN Power domain A17 slave core 1 power down enable 1'b0: power on 1'b1: power off
0	RW	0x0	PD_A17_0_DWN_EN Power Domain A17 primary core power down enable 1'b0: power on 1'b1: power off

**PMU\_PWRDN\_ST**

Address: Operational Base + offset (0x000c)

System power gating status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13	RW	0x0	I2_standywfi
12	RW	0x0	I2_flush_done
11	RO	0x0	pd_scu_pwr_st Power domain SCU power status 1'b0: power on 1'b1: power off
10	RO	0x0	pd_hevc_pwr_st Power domain HEVC power status 1'b0: power on 1'b1: power off
9	RO	0x0	pd_gpu_pwr_st Power domain GPU power status 1'b0: power on 1'b1: power off
8	RO	0x0	pd_video_pwr_st Power domain VIDEO power status 1'b0: power on 1'b1: power off

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RO	0x0	pd_vio_pwr_st Power domain VIO power status 1'b0: power on 1'b1: power off
6	RO	0x0	pd_peri_pwr_st Power domain PERI power status 1'b0: power on 1'b1: power off
5	RO	0x0	pd_bus_pwr_st Power domain BUS power status 1'b0: power on 1'b1: power off
4	RO	0x0	reserved
3	RW	0x0	pd_A17_3_pwr_st Power domain A17 slave core 3 power status 1'b0: power on 1'b1: power off
2	RW	0x0	pd_A17_2_pwr_st Power domain A17 slave core 2 power status 1'b0: power on 1'b1: power off
1	RW	0x0	pd_A17_1_pwr_st Power domain A17 slave core 1 power status 1'b0: power on 1'b1: power off
0	RW	0x0	pd_A17_0_pwr_st Power domain A17 primary core power status 1'b0: power on 1'b1: power off

**PMU\_IDLE\_REQ**

Address: Operational Base + offset (0x0010)

PMU Noc idle req control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9	RW	0x0	idle_req_hevc_cfg software config HEVC domain flush transaction request 1'b1: idle req 1'b0: not idle req
8	RW	0x0	idle_req_cpup_cfg software config CPUP domain flush transaction request 1'b1: idle req 1'b0: not idle req

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	idle_req_dma_cfg software config DMA domain flush transaction request 1'b1: idle req 1'b0: not idle req
6	RW	0x0	idle_req_alive_cfg software config ALIVE domain flush transaction request 1'b1: idle req 1'b0: not idle req
5	RW	0x0	idle_req_core_cfg software config CORE domain flush transaction request 1'b1: idle req 1'b0: not idle req
4	RW	0x0	idle_req_vio_cfg software config VIO domain flush transaction request 1'b1: idle req 1'b0: not idle req
3	RW	0x0	idle_req_video_cfg software config VIDEO domain flush transaction request 1'b1: idle req 1'b0: not idle req
2	RW	0x0	idle_req_gpu_cfg software config GPU domain flush transaction request 1'b1: idle req 1'b0: not idle req
1	RW	0x0	idle_req_peri_cfg software config PERI domain flush transaction request 1'b1: idle req 1'b0: not idle req
0	RW	0x0	idle_req_bus_cfg software config BUS domain flush transaction request 1'b1: idle req 1'b0: not idle req

**PMU\_IDLE\_ST**

Address: Operational Base + offset (0x0014)

PMU Noc idle status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
25	RW	0x0	idle_ack_hevc hevc domain flush transaction acknowledge 1'b0: no ack 1'b1: ack
24	RW	0x0	idle_ack_cpup cpup domain flush transaction acknowledge 1'b0: no ack 1'b1: ack
23	RW	0x0	idle_ack_dma dma domain flush transaction acknowledge 1'b0: no ack 1'b1: ack
22	RW	0x0	idle_ack_alive ALIVE domain flush transaction acknowledge 1'b0: no ack 1'b1: ack
21	RW	0x0	idle_ack_core core domain flush transaction acknowledge 1'b0: no ack 1'b1: ack
20	RW	0x0	idle_ack_vio VIO domain flush transaction acknowledge 1'b0: no ack 1'b1: ack
19	RW	0x0	idle_ack_video VIDEO domain flush transaction acknowledge 1'b0: no ack 1'b1: ack
18	RW	0x0	idle_ack_gpu GPU domain flush transaction acknowledge 1'b0: no ack 1'b1: ack
17	RW	0x0	idle_ack_peri PERI domain flush transaction acknowledge 1'b0: no ack 1'b1: ack
16	RW	0x0	idle_ack_bus BUS domain flush transaction acknowledge 1'b0: no ack 1'b1: ack
15:10	RO	0x0	reserved
9	RW	0x0	IDLE_HEVC HEVC domain flush transaction finish(idle) 1'b0: no finish 1'b1: finish

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	RW	0x0	IDLE_CPUP CPUP domain flush transaction finish(idle) 1'b0: no finish 1'b1: finish
7	RW	0x0	IDLE_DMA DMA domain flush transaction finish(idle) 1'b0: no finish 1'b1: finish
6	RW	0x0	IDLE_ALIVE ALIVE domain flush transaction finish(idle) 1'b0: no finish 1'b1: finish
5	RW	0x0	IDLE_CORE CORE domain flush transaction finish(idle) 1'b0: no finish 1'b1: finish
4	RW	0x0	IDLE_VIO VIO domain flush transaction finish(idle) 1'b0: no finish 1'b1: finish
3	RW	0x0	IDLE_VIDEO VIDEO domain flush transaction finish(idle) 1'b0: no finish 1'b1: finish
2	RW	0x0	IDLE_GPU GPU domain flush transaction finish(idle) 1'b0: no finish 1'b1: finish
1	RW	0x0	IDLE_PERI PERI domain flush transaction finish(idle) 1'b0: no finish 1'b1: finish
0	RW	0x0	IDLE_BUS BUS domain flush transaction finish(idle) 1'b0: no finish 1'b1: finish

**PMU\_PWRMODE\_CON**

Address: Operational Base + offset (0x0018)

PMU configuration register in power mode flow

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:23	RO	0x0	reserved
22	RW	0x0	ddr1io_ret_de_req ddr1io retention de-assert request 1'b0: de-assert request 1'b1: not de-assert request

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
21	RW	0x0	ddr0io_ret_de_req ddr0io retention de-assert request 1'b0: de-assert request 1'b1: not de-assert request
20	RW	0x0	ddrc1_gating_en ddrc1 clock auto gating after self-refresh in low power mode 1'b1: auto gating 1'b0: not auto gating
19	RW	0x0	ddr0_gating_en ddrc0 auto gating in low power mode 1'b0: disable 1'b1: enable
18	RW	0x0	ddr1io_ret_en ddr1 io ret enable in low power mode 1'b1: enable 1'b0: disable
17	RW	0x0	ddr0io_ret_en DDR0 IO retention function enable or not 1'b0: DDR0 IO retention disable 1'b1: DDR0 IO retention enable
16	RW	0x0	sref1_enter_en ddr1 enter self-refresh in low power mode 1'b1: enable 1'b0: disable
15	RW	0x0	sref0_enter_en DDR0 enter self-refresh enable in low power mode 1'b0: disable DDR0 enter self-refresh 1'b1: enable DDR0 enter self-refresh
14	RW	0x0	wakeup_reset_en wakeup reset enable if power up 1'b0: diable 1'b1: enable
13	RW	0x0	input_clamp_en input clamp enable if power off input clamp for PD_PMU enable if power off 1'b0: disable 1'b1: enable
12	RW	0x0	osc_24m_dis 24MHz OSC disable in low power mode 1'b0: 24MHz OSC enable 1'b1: 24MHz OSC disable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x0	pmu_use_lf pmu domain clock switch to 32.768kHz enable 1'b0: not switch to 32.768kHz 1'b1: switch to 32.768kHz
10	RW	0x0	alive_use_lf ALIVE domain clock switch to 32.768kHz enable 1'b0: not switch to 32.768kHz 1'b1: switch to 32.768kHz
9	RW	0x0	pwroff_comb three power off signal combination 1'b0: not combine 1'b1: combine enable
8	RW	0x0	chip_pd_en chip power down enable in power mode flow 1'b0: chip power on 1'b1: chip power off
7	RW	0x0	pll_pd_en pll power down enable in power mode flow 1'b0: pll power on 1'b1: pll power off
6	RW	0x0	scu_en scu power down enable in power mode flow 1'b0: scu power on 1'b1: scu power off
5	RW	0x0	A17_0_pd_en A17_0 power down in power mode flow 1'b0: A17_0 power on 1'b1: A17_0 power off
4	RW	0x0	bus_pd_en bus power off enable in low power mode 1'b0: bus power on 1'b1: bus power off
3	RW	0x0	I2flush_en I2 flush enable 1'b1: I2 flush enable 1'b0: I2 flush disable
2	RW	0x0	global_int_disable Global interrupt disable 1'b0: enable global interrupt 1'b1: disable global interrupt

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	clk_core_src_gate_en A17 core clock source gating enable in idle mode 1'b0: enable 1'b1: disable
0	RW	0x0	power_mode_en power mode flow enable 1'b0: disable 1'b1: enable

**PMU\_PWR\_STATE**

Address: Operational Base + offset (0x001c)

PMU Low power mode state

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved
30	RW	0x0	A17_CLK_EN A17 source clock enable 1'b0: state not happened 1'b1: state happened
29	RW	0x0	TRANS_RESTORE noc trans restore 1'b0: state not happened 1'b1: state happened
28	RW	0x0	A17_0_PWRUP A17 core0 power up state 1'b0: state not happened 1'b1: state happened
27	RW	0x0	L2MEM_PWRUP pd_l2mem powerup state 1'b0: state not happened 1'b1: state happened
26	RW	0x0	SREF_EXIT ddr exit self-refresh 1'b0: state not happened 1'b1: state happened
25	RW	0x0	DDR_IO_PWRUP ddr io powerup state 1'b0: state not happened 1'b1: state happened
24	RW	0x0	BUS_PWRUP pd_bus powerup state 1'b0: state not happened 1'b1: state happened

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23	RW	0x0	PLL_PWRUP pll power up state 1'b0: state not happened 1'b1: state happened
22	RW	0x0	WAKEUP_RESET_CLR deassert wakeup reset 1'b0: state not happened 1'b1: state happened
21	RW	0x0	ALIVE_PMU_HF pd_alive & pd_pmu switch to normal clock 1'b0: state not happened 1'b1: state happened
20	RW	0x0	X24M_OSC_EN 24MHz OSC enable 1'b0: state not happened 1'b1: state happened
19	RW	0x0	RELEASE_CLAMP release pd_pmu input clamp 1'b0: state not happened 1'b1: state happened
18	RW	0x0	EXT_PWRUP ext pmic power up 1'b0: state not happened 1'b1: state happened
17	RW	0x0	WAKEUP_RESET wakeup reset 1'b0: state not happened 1'b1: state happened
16	RW	0x0	WAIT_WAKEUP wait wakeup state 1'b0: state not happened 1'b1: state happened
15	RW	0x0	X24M_OSC_DIS 24MHz soc disable state 1'b0: state not happened 1'b1: state happened
14	RW	0x0	POWEROFF chip power off state 1'b0: state not happened 1'b1: state happened
13	RW	0x0	INPUT_CLAMP pd_pmu input clamp 1'b0: state not happened 1'b1: state happened

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x0	PLL_PWRDN pll power down state 1'b0: state not happened 1'b1: state happened
11	RW	0x0	ALIVE_PMU_LF pd_alive&pd_pmu switch to 32khz 1'b0: state not happened 1'b1: state happened
10	RW	0x0	L2MEM_PWRDN l2 mem power down state 1'b0: state not happened 1'b1: state happened
9	RW	0x0	A17_0_PWRDN A17 core0 power down state 1'b0: state not happened 1'b1: state happened
8	RW	0x0	BUS_PWRDN pd_bus power down state 1'b0: A17_0 power on 1'b1: A17_0 power off
7	RW	0x0	DDR_IO_PWROFF ddr io power off 1'b0: state not happened 1'b1: state happened
6	RW	0x0	DDR_IO_RET DDR io retention 1'b0: state not happened 1'b1: state happened
5	RW	0x0	SREF_ENTER ddr selfrefresh enter 1'b0: state not happened 1'b1: state happened
4	RW	0x0	TRANS_NO_FIN transfer no finish 1'b0: state not happened 1'b1: state happened
3	RW	0x0	A17_CLK_DIS A17 clock disable 1'b0: state not happened 1'b1: state happened
2	RW	0x0	STANDBYL2 L2 Standby 1'b0: state not happened 1'b1: state happened

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	L2FLUSH_REQ L2 Flush req 1'b0: state not happened 1'b1: state happened
0	RW	0x0	NORMAL normal state 1'b0: state not happened 1'b1: state happened

**PMU\_OSC\_CNT**

Address: Operational Base + offset (0x0020)

24MHz OSC stabilization counter threshold

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19:0	RW	0x05dc0	osc_stabl_cnt_thresh 24MHz OSC stabilization counter threshold

**PMU\_PLL\_CNT**

Address: Operational Base + offset (0x0024)

PLL lock counter threshold

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RW	0x100	pllrst_cnt_thresh PLL reset wait counter threshold
19:0	RW	0x04000	plllock_cnt_thresh PLL lock wait counter threshold

**PMU\_STABL\_CNT**

Address: Operational Base + offset (0x0028)

External PMU stabilization counter threshold

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19:0	RW	0x05dc0	pmu_stabl_cnt_thresh External PMU stabilization counter threshold

**PMU\_DDR0IO\_PWRON\_CNT**

Address: Operational Base + offset (0x002c)

DDR0 IO power on counter threshold

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19:0	RW	0x05dc0	ddr0io_pwron_cnt_thresh DDR0 IO power on counter threshold

**PMU\_DDR1IO\_PWRON\_CNT**

Address: Operational Base + offset (0x0030)

DDR1 IO power on counter threshold

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19:0	RW	0x05dc0	ddr1io_pwron_cnt_thresh DDR1 IO power on counter threshold

**PMU\_CORE\_PWRDWN\_CNT**

Address: Operational Base + offset (0x0034)

CORE domain power down waiting counter in sleep mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19:0	RW	0x05dc0	core_pwrdown_cnt_thresh CORE domain power down waiting counter threshold

**PMU\_CORE\_PWRUP\_CNT**

Address: Operational Base + offset (0x0038)

CORE domain power up waiting counter in sleep mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19:0	RW	0x05dc0	core_pwrup_cnt_thresh CORE domain power up waiting counter threshold

**PMU\_GPU\_PWRDWN\_CNT**

Address: Operational Base + offset (0x003c)

GPU domain power down waiting counter in sleep mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19:0	RW	0x05dc0	gpu_pwrdown_cnt_thresh GPU domain power down waiting counter threshold

**PMU\_GPU\_PWRUP\_CNT**

Address: Operational Base + offset (0x0040)

GPU domain power up waiting counter in sleep mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19:0	RW	0x05dc0	gpu_pwrup_cnt_thresh GPU domain power up waiting counter threshold

**PMU\_WAKEUP\_RST\_CLR\_CNT**

Address: Operational Base + offset (0x0044)

Wakeup reset deassert state wait counter in power off mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19:0	RW	0x05dc0	wakeup_RST_CLR_CNT_THRESH Power off mode wakeup reset clear counter threshold

**PMU\_SFT\_CON**

Address: Operational Base + offset (0x0048)

PMU Software control in normal mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15	RW	0x0	I2flush_cfg I2 flush config in normal mode 1'b1: I2 flush req 1'b0: I2 flush disable
14	RW	0x0	osc_disable_cfg software config OSC disable 1'b1: OSC disable 1'b0: OSC enable
13	RW	0x0	osc_bypass osc bypass control 1'b0: disable 1'b1: enable
12	RW	0x0	alive_if_ena_cfg software config ALIVE domain clock switch to 32.768kHz 1'b1: switch to 32.768kHz 1'b0: not switch
11	RW	0x0	pmu_if_ena_cfg software config PMU domain clock switch to 32.768kHz 1'b1: switch to 32.768kHz 1'b0: not switch
10	RW	0x0	power_off_ddr1io_cfg software config power off DDR1 IO 1'b1: power off 1'b0: not power off
9	RW	0x0	ddr1_io_ret_cfg software config DDR1 IO retention 1'b1: retention 1'b0: not retention
8	RW	0x0	upctl1_c_sysreq_cfg software config enter DDR1 self-refresh by lowpower interface 1'b1: request enter self-refresh 1'b0: not enter self-refresh
7	RW	0x0	power_off_ddr0io_cfg software config power off DDR0 IO 1'b1: power off 1'b0: not power off
6	RW	0x0	ddr0_io_ret_cfg software config DDR0 IO retention 1'b1: retention 1'b0: not retention

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	upctl0_c_sysreq_cfg software config enter DDR0 self-refresh by lowpower interface 1'b1: request enter self-refresh 1'b0: not enter self-refresh
4	RW	0x0	clk_core_src_gating_cfg software config A17 core clock source gating 1'b1: gating 1'b0: not gating
3	RW	0x0	dbgnopwrdsn3_enable ARM CORE3 DBGNOPWRDWN function support enable 1'b0: not support 1'b1: support
2	RW	0x0	dbgnopwrdsn2_enable ARM CORE2 DBGNOPWRDWN function support enable 1'b0: not support 1'b1: support
1	RW	0x0	dbgnopwrdsn1_enable ARM CORE1 DBGNOPWRDWN function support enable 1'b0: not support 1'b1: support
0	RW	0x0	dbgnopwrdsn0_enable ARM CORE0 DBGNOPWRDWN function support enable 1'b0: not support 1'b1: support

**PMU\_DDR\_SREF\_ST**

Address: Operational Base + offset (0x004c)

PMU DDR self refresh status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3	RW	0x0	upctl0_c_sysack DDR0 enter self-refresh acknowledge 1'b0: no ack 1'b1: ack
2	RW	0x0	upctl0_c_active DDR0 enter self-refresh 1'b0: no active 1'b1: active

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	upctl1_c_sysack DDR1 enter self-refresh acknowledge 1'b0: no ack 1'b1: ack
0	RW	0x0	upctl1_c_active DDR1 enter self-refresh 1'b0: no active 1'b1: active

**PMU\_INT\_CON**

Address: Operational Base + offset (0x0050)

PMU interrupt configuration register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27	RW	0x0	pd_mem_int_en Power domain L2 MEM power switch interrupt enable 1'b0: disable 1'b1: enable
26	RW	0x0	pd_hevc_int_en Power domain hevc power switch interrupt enable 1'b0: disable 1'b1: enable
25	RW	0x0	pd_gpu_int_en Power domain GPU power switch interrupt enable 1'b0: disable 1'b1: enable
24	RW	0x0	pd_video_int_en Power domain VIDEO power switch interrupt enable 1'b0: disable 1'b1: enable
23	RW	0x0	pd_vio_int_en Power domain VIO power switch interrupt enable 1'b0: disable 1'b1: enable
22	RW	0x0	pd_peri_int_en Power domain PERI power switch interrupt enable 1'b0: disable 1'b1: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
21	RW	0x0	pd_bus_int_en Power domain BUS power switch interrupt enable 1'b0: disable 1'b1: enable
20	RO	0x0	reserved
19	RW	0x0	pd_a2_3_int_en Power domain A17 slave core 3 power switch interrupt enable 1'b0: disable 1'b1: enable
18	RW	0x0	pd_A17_2_int_en Power domain A17 slave core 2 power switch interrupt enable 1'b0: disable 1'b1: enable
17	RW	0x0	pd_A17_1_int_en Power domain A17 slave core 1 power switch interrupt enable 1'b0: disable 1'b1: enable
16	RW	0x0	pd_A17_0_int_en Power domain A17 primary core power switch interrupt enable 1'b0: disable 1'b1: enable
15:6	RO	0x0	reserved
5	RW	0x0	pwrmode_wakeup_int_en wakeup interrupt enable in power mode 1'b0: disable 1'b1: enable
4	RW	0x0	gpoint_wakeup_int_en gpio interrupt wakeup interrupt enable 1'b0: disable 1'b1: enable
3	RW	0x0	sdmmc0_wakeup_int_en SDMMC0 wakeup status interrupt enable 1'b0: disable 1'b1: enable
2	RW	0x0	gpio_wakeup_int_en GPIO0 wakeup status interrupt enable 1'b0: disable 1'b1: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	armint_wakeup_int_en ARM interrupt wakeup status interrupt enable 1'b0: disable 1'b1: enable
0	RW	0x0	pmu_int_en PMU interrupt enable 1'b0: disable 1'b1: enable

**PMU\_INT\_ST**

Address: Operational Base + offset (0x0054)

PMU interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27	RW	0x0	pd_mem_int_st Power domain I2 mem power switch status 1'b0: no power switch happen 1'b1: power switch happen
26	RW	0x0	pd_hevc_int_st Power domain HEVC power switch status 1'b0: no power switch happen 1'b1: power switch happen
25	W1C	0x0	pd_gpu_int_st Power domain GPU power switch status 1'b0: no power switch happen 1'b1: power switch happen
24	W1C	0x0	pd_video_int_st Power domain VIDEO power switch status 1'b0: no power switch happen 1'b1: power switch happen
23	W1C	0x0	pd_vio_int_st Power domain VIO power switch status 1'b0: no power switch happen 1'b1: power switch happen
22	W1C	0x0	pd_peri_int_st Power domain PERI power switch status 1'b0: no power switch happen 1'b1: power switch happen
21	W1C	0x0	pd_bus_int_st Power domain BUS power switch status 1'b0: no power switch happen 1'b1: power switch happen
20	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19	RW	0x0	pd_A17_3_int_st Power domain A17 slave core 3 power switch status 1'b0: no power switch happen 1'b1: power switch happen
18	RW	0x0	pd_A17_2_int_st Power domain A17 slave core 2 power switch status 1'b0: no power switch happen 1'b1: power switch happen
17	RW	0x0	pd_A17_1_int_st Power domain A17 slave core 1 power switch status 1'b0: no power switch happen 1'b1: power switch happen
16	RW	0x0	pd_A17_0_int_st Power domain A17 primary core power switch status 1'b0: no power switch happen 1'b1: power switch happen
15:5	RO	0x0	reserved
4	RW	0x0	pwrmode_wakeup_event_trig power mode flow wakeup 1'b0: no wakeup 1'b1: wakeup
3	RW	0x0	gpoint_wakeup_event_trig ARM interrupt wake-up enent trigger 1'b0: no wakeup 1'b1: wakeup
2	W1C	0x0	sdmmc0_wakeup_event_trig SDMMC0 wake-up enent trigger 1'b0: no wakeup 1'b1: wakeup
1	W1C	0x0	gpio_wakeup_event_trig GPIO0 wake-up enent trigger 1'b0: no wakeup 1'b1: wakeup
0	RW	0x0	armint_wakeup_event_trig ARM interrupt wake-up enent trigger 1'b0: no wakeup 1'b1: wakeup

**PMU\_BOOT\_ADDR\_SEL**

Address: Operational Base + offset (0x0058)

boot\_addr\_sel in power mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00005dc0	boot_addr_sel boot addr sel when wakeup from power mode

**PMU\_GRF\_CON**

Address: Operational Base + offset (0x005c)

grf control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:10	RW	0x00	GRF_NPOR_CRNT_CTRL Npor signal crnt control register
9:4	RW	0x00	GRF_TEST_CRNT_CTRL Test signal crnt control register
3:2	RW	0x2	GRF_X32K_CRNT_CTRL X32K signal crnt control register
1:0	RW	0x0	GRF_X24M_CRNT_CTRL X24M signal crnt control register

**PMU\_GPIO\_SR**

Address: Operational Base + offset (0x0060)

GPIO slew rate control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:19	RO	0x0	reserved
18	RW	0x0	gpio0_c2_sr gpio0_c2 slew rate control 1'b0: slow (half frequency) 1'b1: fast
17	RW	0x1	gpio0_c1_sr gpio0_c1 slew rate control 1'b0: slow (half frequency) 1'b1: fast
16	RW	0x0	gpio0_c0_sr gpio0_c0 slew rate control 1'b0: slow (half frequency) 1'b1: fast
15	RW	0x0	gpio0_b7_sr gpio0_b7 slew rate control 1'b0: slow (half frequency) 1'b1: fast
14	RW	0x0	gpio0_b6_sr gpio0_b6 slew rate control 1'b0: slow (half frequency) 1'b1: fast
13	RW	0x0	gpio0_b5_sr gpio0_b5 slew rate control 1'b0: slow (half frequency) 1'b1: fast

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x0	gpio0_b4_sr gpio0_b4 slew rate control 1'b0: slow (half frequency) 1'b1: fast
11	RW	0x0	gpio0_b3_sr gpio0_b3 slew rate control 1'b0: slow (half frequency) 1'b1: fast
10	RW	0x0	gpio0_b2_sr gpio0_b2 slew rate control 1'b0: slow (half frequency) 1'b1: fast
9	RW	0x0	gpio0_b1_sr gpio0_b1 slew rate control 1'b0: slow (half frequency) 1'b1: fast
8	RW	0x0	gpio0_b0_sr gpio0_b0 slew rate control 1'b0: slow (half frequency) 1'b1: fast
7	RW	0x0	gpio0_a7_sr gpio0_a7 slew rate control 1'b0: slow (half frequency) 1'b1: fastII
6	RW	0x0	gpio0_a6_sr gpio0_a6 slew rate control 1'b0: slow (half frequency) 1'b1: fast
5	RW	0x0	gpio0_a5_sr gpio0_a5 slew rate control 1'b0: slow (half frequency) 1'b1: fast
4	RW	0x0	gpio0_a4_sr gpio0_a4 slew rate control 1'b0: slow (half frequency) 1'b1: fast
3	RW	0x0	gpio0_a3_sr gpio0_a3 slew rate control 1'b0: slow (half frequency) 1'b1: fast
2	RW	0x0	gpio0_a2_sr gpio0_a2 slew rate control 1'b0: slow (half frequency) 1'b1: fast

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	gpio0_a1_sr gpio0_a1 slew rate control 1'b0: slow (half frequency) 1'b1: fast
0	RW	0x0	gpio0_a0_sr gpio0_a0 slew rate control 1'b0: slow (half frequency) 1'b1: fast

**PMU\_GPIO0\_A\_PULL**

Address: Operational Base + offset (0x0064)

GPIO0A input to PU/PD programmation section

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:14	RW	0x1	gpio0_a7_pull gpio0_a7 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
13:12	RW	0x1	gpio0_a6_pull gpio0_a6 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
11:10	RW	0x1	gpio0_a5_pull gpio0_a5 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
9:8	RW	0x1	gpio0_a4_pull gpio0_a4 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:6	RW	0x1	gpio0_a3_pull gpio0_a3 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
5:4	RW	0x1	gpio0_a2_pull gpio0_a2 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
3:2	RW	0x2	gpio0_a1_pull gpio0_a1 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
1:0	RW	0x2	gpio0_a0_pull gpio0_a0 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)

**PMU\_GPIO0\_B\_PULL**

Address: Operational Base + offset (0x0068)

GPIO0B input to PU/PD programmation section

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:14	RW	0x1	gpio0_b7_pull gpio0_b7 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13:12	RW	0x1	gpio0_b6_pull gpio0_b6 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
11:10	RW	0x1	gpio0_b5_pull gpio0_b5 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
9:8	RW	0x1	gpio0_b4_pull gpio0_b4 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
7:6	RW	0x1	gpio0_b3_pull gpio0_b3 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
5:4	RW	0x1	gpio0_b2_pull gpio0_b2 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
3:2	RW	0x1	gpio0_b1_pull gpio0_b1 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RW	0x1	gpio0_b0_pull gpio0_b0 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)

**PMU\_GPIO0\_C\_PULL**

Address: Operational Base + offset (0x006c)

GPIO0C input to PU/PD programmation section

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:4	RW	0x1	gpio0_c2_pull gpio0_c2 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
3:2	RW	0x1	gpio0_c1_pull gpio0_c1 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)
1:0	RW	0x1	gpio0_c0_pull gpio0_c0 pu/pd programmation section [p2:p1] 2'b00: Z(Normal operation) 2'b01: weak 1 (pull-up) 2'b10: weak 0 (pull-down) 2'b11: repeater (bus keeper)

**PMU\_GPIO0\_A\_DRV**

Address: Operational Base + offset (0x0070)

GPIO0A Drive strength slector

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:14	RW	0x1	gpio0_a7_e gpio0_a7 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13:12	RW	0x1	gpio0_a6_e gpio0_a6 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
11:10	RW	0x1	gpio0_a5_e gpio0_a5 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
9:8	RW	0x1	gpio0_a4_e gpio0_a4 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
7:6	RW	0x1	gpio0_a3_e gpio0_a3 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
5:4	RW	0x1	gpio0_a2_e gpio0_a2 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
3:2	RW	0x2	gpio0_a1_e gpio0_a1 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RW	0x2	gpio0_a0_e gpio0_a0 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA

**PMU\_GPIO0\_B\_DRV**

Address: Operational Base + offset (0x0074)

GPIO0B Drive strength slector

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:14	RW	0x1	gpio0_b7_e gpio0_b7 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
13:12	RW	0x1	gpio0_b6_e gpio0_b6 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
11:10	RW	0x1	gpio0_b5_e gpio0_b5 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
9:8	RW	0x1	gpio0_b4_e gpio0_b4 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:6	RW	0x1	gpio0_b3_e gpio0_b3 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
5:4	RW	0x1	gpio0_b2_e gpio0_b2 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
3:2	RW	0x1	gpio0_b1_e gpio0_b1 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
1:0	RW	0x1	gpio0_b0_e gpio0_b0 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA

**PMU\_GPIO0\_C\_DRV**

Address: Operational Base + offset (0x0078)

GPIO0C Drive strength slector

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:4	RW	0x1	gpio0_c2_e gpio0_c2 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:2	RW	0x1	gpio0_c1_e gpio0_c1 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA
1:0	RW	0x1	gpio0_c0_e gpio0_c0 drive strength slector [e2:e1] 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA

**PMU\_GPIO\_OP**

Address: Operational Base + offset (0x007c)

GPIO0 output value

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:19	RO	0x0	reserved
18	RW	0x0	gpio0_c2_op gpio0_c2 output value
17	RW	0x0	gpio0_c1_op gpio0_c1 output value
16	RW	0x0	gpio0_c0_op gpio0_c0 output value
15	RW	0x0	gpio0_b7_op gpio0_b7 output value
14	RW	0x0	gpio0_b6_op gpio0_b6 output value
13	RW	0x0	gpio0_b5_op gpio0_b5 output value
12	RW	0x0	gpio0_b4_op gpio0_b4 output value
11	RW	0x0	gpio0_b3_op gpio0_b3 output value
10	RW	0x0	gpio0_b2_op gpio0_b2 output value
9	RW	0x0	gpio0_b1_op gpio0_b1 output value
8	RW	0x0	gpio0_b0_op gpio0_b0 output value
7	RW	0x0	gpio0_a7_op gpio0_a7 output value
6	RW	0x0	gpio0_a6_op gpio0_a6 output value

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	gpio0_a5_op gpio0_a5 output value
4	RW	0x0	gpio0_a4_op gpio0_a4 output value
3	RW	0x0	gpio0_a3_op gpio0_a3 output value
2	RW	0x0	gpio0_a2_op gpio0_a2 output value
1	RW	0x0	gpio0_a1_op gpio0_a1 output value
0	RW	0x0	gpio0_a0_op gpio0_a0 output value

**PMU\_GPIO0\_SEL18**

Address: Operational Base + offset (0x0080)

gpio0 1.8v/3.3v sel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2	RW	0x1	gpio0_c0_smt
1	RW	0x1	gpio0_b7_smt gpio0_a0 output value
0	RW	0x0	gpio0_a0_op gpio0_a0 output value 1'b0: >=2.5v 1'b1: <=1.8v

**PMU\_GPIO0\_A\_IOMUX**

Address: Operational Base + offset (0x0084)

GPIO0A iomux sel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:6	RW	0x0	gpio0_a3 iomux 1'b0: gpioa3 1'b1: ddr1_retention
5	RO	0x0	reserved
4	RW	0x0	gpio0_a2 iomux 1'b0: gpioa2 1'b1: ddr0_retention
3	RO	0x0	reserved
2	RW	0x0	gpio0_a1 iomux 1'b0: gpioa1 1'b1: ddrio_pwroff
1	RO	0x0	reserved
0	RW	0x0	gpio0_a0 iomux 1'b0: gpioa0 1'b1: global_pwroff

**PMU\_GPIO0\_B\_IOMUX**

Address: Operational Base + offset (0x0088)

GPIO0B iomux sel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	RO	0x0	reserved
14	RW	0x0	gpio0_b7 iomux 1'b0: gpiob7 1'b1: i2c0pmu_sda
13:11	RO	0x0	reserved
10	RW	0x0	gpio0_b5 iomux 1'b0: gpiob5 1'b1: CLK_27M
9:5	RO	0x0	reserved
4	RW	0x0	gpio0_b2 iomux 1'b0: gpiob2 1'b1: tsadc_int
3:0	RO	0x0	reserved

**PMU\_GPIO0\_C\_IOMUX**

Address: Operational Base + offset (0x008c)

GPIO0C iomux sel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:2	RW	0x0	gpio0_c1 iomux 2'b00: gpioc1 2'b01: test_clkout 2'b10: clkt1_27m 2'b11: reserved
1	RO	0x0	reserved
0	RW	0x0	gpio0_c0 iomux 1'b0: gpioc0 1'b1: i2c0pmu_scl

**PMU\_PWRMODE\_CON1**

Address: Operational Base + offset (0x0090)

PMU PowerMode CON1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	-	0x0	reserved
9	RW	0x0	clr_vio issue idle_req_vio in low power mode 1'b0: not issue 1'b1: issue
8	RW	0x0	clr_hevc issue idle_req_hevc in low power mode 1'b0: not issue 1'b1: issue

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	clr_video issue idle_req_video in low power mode 1'b0: not issue 1'b1: issue
6	RW	0x0	clr_gpu issue idle_req_gpu in low power mode 1'b0: not issue 1'b1: issue
5	RW	0x0	clr_peri issue idle_req_peri in low power mode 1'b0: not issue 1'b1: issue
4	RW	0x0	clr_dma issue idle_req_dma in low power mode 1'b0: not issue 1'b1: issue
3	RW	0x0	clr_alive issue idle_req_alive in low power mode 1'b0: not issue 1'b1: issue
2	RW	0x0	clr_cpup issue idle_req_cpup in low power mode 1'b0: not issue 1'b1: issue
1	RW	0x0	clr_core issue idle_req_core in low power mode 1'b0: not issue 1'b1: issue
0	RW	0x0	clr_bus issue idle_req_bus in low power mode 1'b0: not issue 1'b1: issue

**PMU\_SYS\_REG0**

Address: Operational Base + offset (0x0094)

PMU system register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	pmu_sys_reg0 PMU system register0

**PMU\_SYS\_REG1**

Address: Operational Base + offset (0x0098)

PMU system register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	pmu_sys_reg1 PMU system register1

**PMU\_SYS\_REG2**

Address: Operational Base + offset (0x009c)

PMU system register2

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	pmu_sys_reg2 PMU system register2

**PMU\_SYS\_REG3**

Address: Operational Base + offset (0x00a0)

PMU system register3

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	pmu_sys_reg3 PMU system register3

## 4.6 Timing Diagram

### 4.6.1 Each domain power switch timing

The following figure is the each domain power down and power up timing.

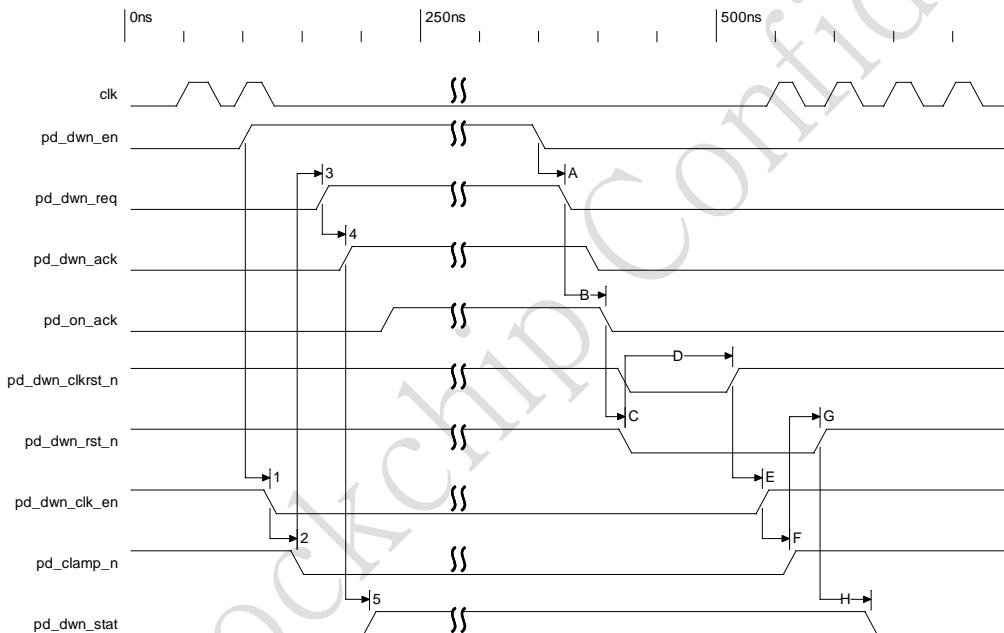


Fig. 4-3 Each Domain Power Switch Timing

### 4.6.2 External wakeup PAD timing

The PMU supports a lot of external wakeup sources, such as SD/MMDC, USBDEV, SIM0/1 detect wakeup, GPIO0 wakeup source and so on. All these external wakeup sources must meet the timing requirement (at least 200us) when the wakeup event is asserted. The following figure gives the timing information.

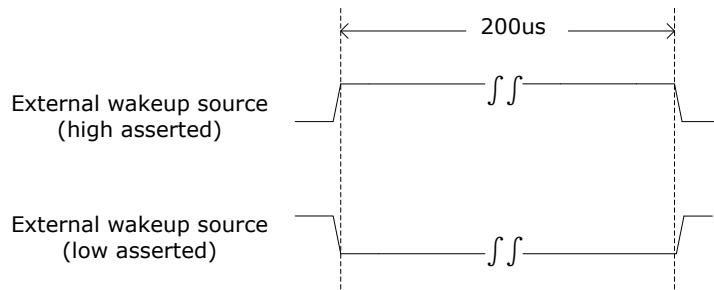


Fig. 4-4 External Wakeup Source PAD Timing

## 4.7 Application Notes

### 4.7.1 Recommend configurations for power mode.

The PMU is a design with great flexibilities, but just for facilities and inheritances, a group of recommend configurations will be shown below for software. And for convenience, we will define several modes.

The RK3288 can support following 5 recommended power modes:

- normal
- idle mode
- deep idle mode
- sleep mode
- power off mode

The following table lists the detailed description of the modes.

Table 4-5 Power Domain Status Summary in all Work Mode

Power Domain	Power Mode				
	Mode0(normal)	Mode1(idle)	Mode2(deepidle)	Mode3(sleep)	Mode4(poweroff)
VD_CORE	PD_A17_0	Running	Standby	Power off	Power off
	PD_A17_1	Running/Standby/Power off	Running/Standby/Power off	Power off	Power off
	PD_A17_2	Running/Standby/Power off	Running/Standby/Power off	Power off	Power off
	PD_A17_3	Running/Standby/Power off	Running/Standby/Power off	Power off	Power off
	PD_SCU	Running	Running	Power off	Power off
	PD_CS	Power on	Power off	Running	Power off
PD_BUS	PD_MEM	Runing	Runing	Power off	Power off
	PD_BUS	Power on	Power on	Running	Power off
	PD_PERI	Power on	Power on/off	Power on/off	Power off
	PD_VIO	Power on	Power on/off	Power on/off	Power off
	PD_VIDEO	Power on	Power on/off	Power on/off	Power off
	PD_HEVC	Power on	Power on/off	Power off	Power off
PD_ALIVE	PD_GPU	Power on	Power on/off	Power on/off	Power off
	PD_ALIVE	Power on	Power on	Power on, Clocked by 24MHz or 32KHz	Power off
	PD_PMU	Power on	Power on	Power on, Clocked by 24MHz or 32KHz	Power on, Clocked by 32KHz
	PLL	All PLLs on	All PLLs on	All PLLs on	All PLLs off
	OSC_24MHz	OSC enable	OSC enable	OSC enable	OSC enable/disable
	DDR	Running	Running	Self refresh	Self refresh
Wakeup Sources		1. all arm interrupts (include EVENT1 input) 2. sdmmc0 detect_n 3. gpio int(not gpio0) 4. gpio0 io	1. all arm interrupts (include EVENT1 input) 2. sdmmc0 detect_n 3. gpio int(not gpio0) 4. gpio0 io	1. sdmmc0 detect_n 2. gpio io int(not gpio0) 3. gpio0 io	GPIO0 IO
		Software control to wake up all the module in power off or clock off states			

### Normal mode

In this mode, you can power off/on or enable/disable the following power domain to save power: PD\_PERI/PD\_VIO/PD\_VIDEO/PD\_HEVC/PD\_GPU

### Idle mode

This mode is used when the core do not have load for a shot while such as waiting for interrupt and the software want to save power by gating Cortex-A17 source clock.

In idle mode, core1/2/3 of Cortex-A17 should be either power off or in WFI/WFE state. The

core0 of A17 should be in WFI/WFE state. The configurations of core clock source gating and disable global interrupt are presented. The Cortex-A17 can waked up by an interrupt.

### **Deep idle mode**

Deep idle mode is used in the scenario of audio player. In deep idle mode, powering off Cortex-A17 cores or VD\_CORE voltage domain is operational, and others are same as normal mode.

In deep idle mode, you can set ddr enter the self-refresh, and power off DDRIO and enable DDR retention function in this mode, but it will takes a longer time for the recovery of DDR IO.

### **Sleep mode**

The sleep mode can power off all power domains except PD\_ALIVE. The VD\_CORE is turned off externally, PD\_BUS power off by hardware, and other domains power off by software.

In sleep mode the clock of PD\_ALIVE can be switched from 24MHz to 32.768kHz optionally by hardware.

In sleep mode all PLLs power down mandatorily to save power by hardware.

In sleep mode OSC can be disabled optionally by hardware.

In sleep mode DDR self-refresh can be issued by hardware mandatorily.

In sleep mode DDR IO can power off and enter retention optionally by hardware.

### **Power off mode**

The power off mode turns off the power of all VD\_LOGIC externally.

In power off mode all PLLs power down mandatorily to save power by hardware.

In power off mode OSC disable request should be send by hardware.

In power off mode DDR self-refresh should be issued mandatorily by hardware.

In power off mode DDR IO can power off and enter retention optionally by hardware.

## **4.7.2 System Register**

PMU support 4 words register: PMU\_SYS\_REG0, PMU\_SYS\_REG1, PMU\_SYS\_REG2, PMU\_SYS\_REG3. These registers are always on no matter what low power mode. So software can use these registers to retain some information which is useful after wakeup from any mode.

## **4.7.3 Configuration Constraint**

In order to shut down the power domains correctly, the software must obey the rules bellow:

- Send NIU request to the NIU in power domain that you want to shut down.
- Querying PMU\_NOC\_ST register to get the information until the pacific NIU is in idle state.
- Send power request to the power domain through PMU\_PWRDN\_CON register.
- Querying PMU\_PWRDN\_ST register to make sure the pacific power domain is power down.

The power domains controlled only by software are showing below:

PD\_VIO, PD\_PERI, PD\_GPU, PD\_VIDEO, PD\_HEVC, and PD\_A17\_1 and PD\_A17\_2 to PD\_A17\_3.

So you must power off these power domains before enter low power mode if you need.

## **4.7.4 Poweroff Request Combine**

There is only two poweroff request, one is for VD\_CORE and VD\_LOGIC (power\_off\_req),

another is for DDRIO (power\_off\_ddrio).

### **POWER\_OFF\_REQ**

In normal mode, If you set the chip power down (bit[13] of PMU\_PWRDN\_CON), or set core power down (bit[12] of PMU\_PWRDN\_CON), the power\_off\_req will be set to 1 immediate.

In Low Power mode, if you set the chip power down (bit[8] of PMU\_PWRMODE\_CON) or set core power down (bit[6] of PMU\_PWRMODE\_CON), the power\_off\_req will be set to 1 at the sub-step POWEROFF.

### **POWER\_OFF\_DDRIO**

In normal mode, If you set the power\_off\_ddr0io\_cfg (bit[7] of PMU\_SFT\_CON), or set power\_off\_ddr1io\_cfg (bit[10] of PMU\_SFT\_CON), the power\_off\_ddrio will be set to 1 immediate.

In Low Power mode, if you set the ddr0io\_ret\_en (bit[17] of PMU\_PWRMODE\_CON) or set ddr1io\_ret\_en (bit[18] of PMU\_PWRMODE\_CON), the power\_off\_ddrio will be set to 1 at the sub-step DDR\_IO\_POWEROFF.

In Low Power mode, if you set pwroff\_comb (bit[9] of PMU\_PWRMODE\_CON) at same time, the power\_off\_ddrio would not be set to 1, and the power\_off\_req would be set to 1 at the sub-step DDR\_IO\_POWEROFF.

## Chapter 5 System Debug

### 5.1 Overview

The RK3288 uses the CoreSight Technology to support real-time debug access and trace for the multi-core. A standard infrastructure is implemented for the capture and transmission of trace data, combination of multiple data streams by funneling together, and then output of data to a trace port.

#### 5.1.1 Features

- Invasive debug with core halted
- cross-triggering, the ECT provides a standard interconnect mechanism to pass debug or profiling events around the SOC
- Trace, capture and transmission trace data using PTM and TPIU
- Real-time access system memory and peripheral register without halting the CPU, using DAP AHB master

#### 5.1.2 Debug components address map

The following table shows the debug components address in memory map:

Module	Base Address	Size
DAP_ROM	0xFFB80000	4KB
CPU_ROM	0xFFBA0000	4KB
CPUDBG0	0xFFBB0000	4KB
CPUPMU0	0xFFBB1000	4KB
CPUDBG1	0xFFBB2000	4KB
CPUPMU1	0xFFBB3000	4KB
CPUDBG2	0xFFBB4000	4KB
CPUPMU2	0xFFBB5000	4KB
CPUDBG3	0xFFBB6000	4KB
CPUPMU3	0xFFBB7000	4KB
CTI0	0xFFBB8000	4KB
CTI1	0xFFBB9000	4KB
CTI2	0xFFBBA000	4KB
CTI3	0xFFBBB000	4KB
PTM0	0xFFBBC000	4KB
PTM1	0xFFBBD000	4KB
PTM2	0xFFBBE000	4KB
PTM3	0xFFBBF000	4KB
Trace Funnel	0xFFBC3000	4KB
CTI(for TPIU)	0xFFBC4000	4KB
Timestamp	0xFFBC5000	4KB
TPIU	0xFFBC6000	4KB

## 5.2 Block Diagram

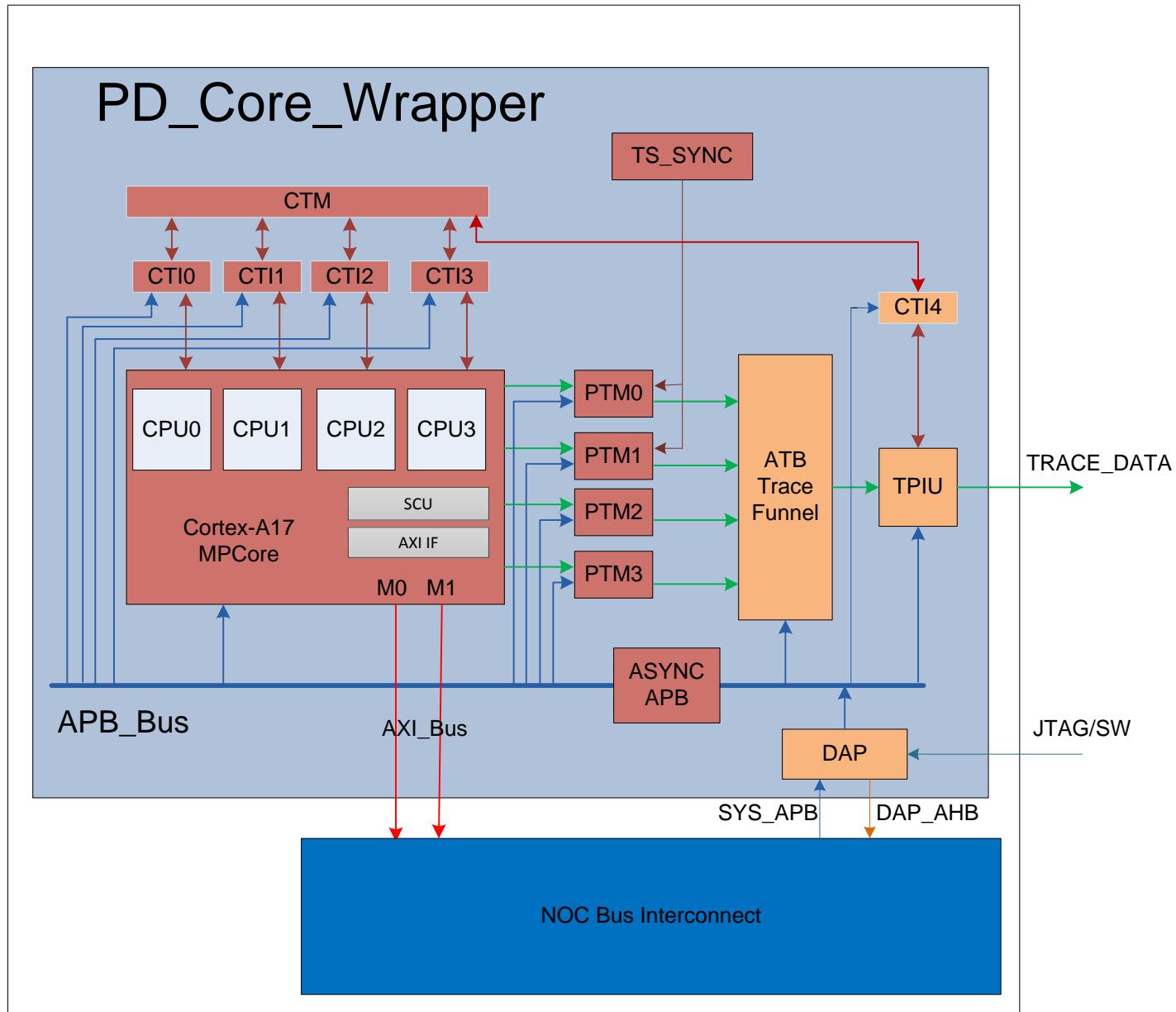


Fig. 5-1 RK3288 Debug system structure

## 5.3 Function description

### 5.3.1 DAP

The Debug Access Port (DAP) is an implementation of an ARM Debug Interface comprising a number of components supplied in a single configuration. All the supplied components fit into the various architectural components for Debug Ports (DPs), which are used to access the DAP from an external debugger and Access Ports (APs), to access on-chip system resources.

The RK3288 DAP has following components:

- Serial Wire JTAG Debug Port(SWJ-DP)
- APB Access Port(APB-AP)
- APB-Mux
- AHB Access Port(AHB-AP)
- ROM table

The debug port is the host tools interface to access the DAP-Lite. This interface controls any access ports provided within the DAP-Lite. The DAP-Lite supports a combined debug port which includes both JTAG and Serial Wire Debug(SWD), with a mechanism that supports switching between them.

The APB-AP acts as a bridge between SWJ-DP and APB bus which translate the debug request to APB bus.

The APB-Mux enables external tools and system access to the debug APB. The APB-Mux encapsulates the multiple interfaces into a single deliverable component, enable multi-master access to the debug APB.

The AHB-AP implements the MEM-AP architecture to directly connect to an AHB based memory system. Connection to other memory systems is possible through suitable bridging local.

The DAP provides an internal ROM table connected to the master debug APB port of the APB-Mux. The debug ROM table is loaded at address 0x00000000 and 0x80000000 of this bus and is accessible from both APB-AP and the system APB input. Bit31 of the address bus is not connected to the ROM table, ensuring that both views read the same value. The ROM table stores the locations of the components on the debug APB.

### 5.3.2 PTM

The PTM is a module that performs real-time instruction flow tracing based on the Program Flow Trace (PFT) architecture. The PTM generates information that trace tools use to reconstruct the execution of all or part of a program.

The PFT architecture assumes that the trace tools can access a copy of the code being traced. For this reason, the PTM generates trace only at certain points in program execution, called waypoints. This reduces the amount of trace data generated by the PTM compared to the ETM protocol. Waypoints are changed in the program flow or events, such as an exception. The trace tools use waypoints to follow the flow of program execution.

### 5.3.3 Trace funnel

The CSTF is used when there is more than one trace source. The CSTF combines multiple trace streams onto a single ATB bus.

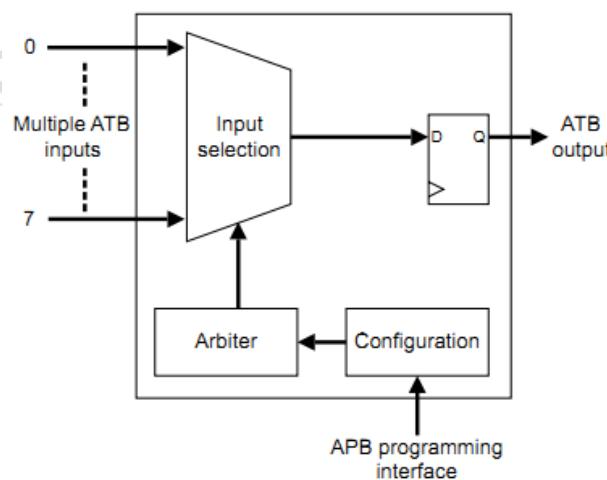


Fig. 5-2 Trace funnel architecture

### 5.3.4 TPIU

The TPIU acts as a bridge between the on-chip trace data, with separate IDs, to a data stream, encapsulating IDs where required, that is then captured by a Trace Port Analyzer (TPA). Fig.

6-1 shows the main blocks of the TPIU and the clock domains.

The TPIU contains the following components:

- Formatter  
Inserts source ID signals into the data packet stream so that trace data can be re-associated with its trace source. See TPIU formatter and FIFO.
- Asynchronous FIFO  
Enables trace data to be driven out at a speed that is not dependent on the on-chip bus clock.
- Register bank  
Contains the management, control and status registers for triggers, flushing behavior and external control.
- Trace out  
The trace out block serializes formatted data before it goes off-chip.
- Pattern Generator  
The pattern generator unit provides a simple set of defined bit sequences or patterns that can be output over the Trace Port and be detected by the TPA or other associated Trace Capture Device (TCD). The TCD can use these patterns to indicate if it is possible to increase or to decrease the trace port clock speed.
- ATB interface  
The TPIU accepts trace data from a trace source, either direct from a trace source or using a Trace Funnel.
- APB interface  
The APB interface is the programming interface for the TPIU.

### 5.3.5 ECT (CTI & CTM)

The ECT for CoreSight consists of a number of CTIs and CTMs connected together. This enables ARM/ETM subsystems to interact. That is cross trigger, with each other. The debug system enables debug support for multiple cores, together with cross triggering between the cores and their respective ETMs.

The main function of the ECT (CTI and CTM) is to pass debug events from one core to another. For example, the ECT can communicate debug state information from one core to another, so that program execution on both processors can be stopped at the same time if required.

- CTI (Cross Trigger Interface)  
The CTI combines and maps the trigger requests, and broadcasts them to all other interfaces on the ECT as channel events. When the CTI receives a channel event it maps this onto a trigger output. This enables subsystems to cross trigger with each other. The receiving and transmitting of triggers is performed through the trigger interface.
- CTM (Cross Trigger Matrix)  
This block controls the distribution of channel events. It provides Channel Interfaces (CIs) for connection to either CTIs or CTMs. This enables multiple CTIs to be linked together.

## 5.4 Register description

### 5.4.1 DAP APB-AP register summary

Name	Offset	Size	Reset Value	Description
DAP_CSW	0x000	W	0x00000002	Control/Status Word, CSW
DAP_TAR	0x004	W	0x00000000	Transfer Address, TAR
DAP_DRW	0x00c	W	NA	Data Read/Write, DRW
DAP_BD0	0x010	W	NA	Bank Data 0, BD0

DAP_BD1	0x014	W	NA	Bank Data 1, BD1
DAP_BD2	0x018	W	NA	Bank Data 2, BD2
DAP_BD3	0x01c	W	NA	Bank Data 3, BD3
DAP_ROM_ADDR	0xf8	W	NA	Debug ROM Address, ROM
DAP_IDR	0xfc	W	0x14770002	Identification Register, IDR

## 5.4.2 DAP APB-AP Detailed Register Description

### DAP\_CSW

Address: APBAP\_BASE + offset(0x000)

Control/Status Word

Bits	Attr	Reset Value	Description
31	RW	0x0	Software access enable. Drives DBGSWENABLE to enable or disable software access to the Debug APB bus in the APB multiplexor. 1'b1: Enable software access 1'b0: Disable software access. Reset value : 1'b0. On exit from reset, defaults to 1'b1 to enable software access.
31:12	RW	0x0	Reserved
11:8	R	0x0	Specifies the mode of operation. 4'b0000: Normal download/upload model 4'b0001-b1111: Reserved Reset value: 4'b0000.
7	R	0x0	Transfer in progress. This field indicates if a transfer is currently in progress on the APB master port.
6	R	0x0	Transfer in progress. This field indicates if a transfer is currently in progress on the APB master port.  Indicates the status of the DEVICEEN input. <ul style="list-style-type: none"><li>If APB-AP is connected to the debug APB, that is, a bus connected only to debug and trace components, it must be permanently enabled by tying DEVICEEN HIGH. This ensures that trace components can still be programmed when DBGEN is LOW. In practice, it is expected that the APB-AP is almost always used in this way.</li><li>If APB-AP is connected to a system APB dedicated to the non-secure world, DEVICEEN must be connected to DBGEN.</li><li>If APB-AP is connected to a system APB dedicated to the secure world, DEVICEEN must be connected to SPIDEN.</li></ul>
5:4	RW	0x0	Auto address increment and packing mode on Read or Write data access. Does not increase if the transaction completes with an error response or the transaction is aborted. Auto address incrementing is not performed on access to banked data registers 0x10-0x1C. The status of these bits is ignored in these cases. 2'b11: Reserved 2'b10: Reserved

			2'b01: Increment 2'b00: Auto increment OFF Increment occurs in word steps. Reset value : 2'b00.
3	R	0x0	Reserved
2:0	R	0x2	Size of the access to perform. Fixed at 3'b010: 32 bits. Reset value: 3'b010.

**DAP\_TAR**

Address: APBAP\_BASE + offset(0x004)

Transfer Address

Bits	Attr	Reset Value	Description
31:2	RW	0x0	Address[31:2] of the current transfer PADDR[31:2]=TAR[31:2] for accesses from Data Read/Write Register at 0x0C. PADDR[31:2]=TAR[31:4]+DAPADDR[3:2] for accesses from Banked Data Registers at 0x10-0x1C and 0x0C.
1:0	R	0x0	Reserved

**DAP\_DRW**

Address: APBAP\_BASE + offset(0x00c)

Data Read/Write

Bits	Attr	Reset Value	Description
31:0	RW	0x0	Write mode: Data value to write for the current transfer. Read mode: Data value read from the current transfer.

**DAP\_BD0-DAP\_BD3**

Address: APBAP\_BASE + offset(0x010) - APBAP\_BASE + offset(0x01c)

Bank Data 0-3

Bits	Attr	Reset Value	Description
31:0	RW	0x0	If DAPADDR[7:4] = 0x0001, so accessing APB-AP registers in the range 0x10-0x1C, then the derived PADDR[31:0] is: <ul style="list-style-type: none"> <li>• Write mode: Data value to write for the current transfer to external address TAR[31:4] + DAPADDR[3:2] + 2'b00.</li> <li>• Read mode: Data value read from the current transfer from external address TAR[31:4] + DAPADDR[3:2] + 2'b00.</li> </ul> Auto address incrementing is not performed on DAP accesses to BD0-BD3. Reset value: 0x00000000

**DAP\_ROM\_ADDR**

Address: APBAP\_BASE + offset(0x0f8)

ROM address

Bits	Attr	Reset Value	Description
31:12	R	0x800000	Base address of the ROM Table. The ROM provides a look-up table of all CoreSight Debug APB components. Read only.

11:0	R	0x000	Read as 0x000 because ROM is present
------	---	-------	--------------------------------------

**DAP\_IDR**

Address: APBAP\_BASE + offset(0x0fc)

Bits	Attr	Reset Value	Description
31:28	R	0x1	Revision. Reset value is 0x1 for APB-AP.
27:24	R	0x4	JEDEC bank. 0x4 indicates ARM Limited.
23:17	R	0x3b	JEDEC code. 0x3B indicates ARM Limited.
16	R	0x1	Memory AP. 0x1 indicates a standard register map is used.
15:8	R	0x00	Reserved
7:0	R	0x02	Identity value. Reset value is 0x02 for APB-AP.

**5.4.3 DAP AHB-AP register summary**

Name	Offset	Size	Reset Value	Description
DAP_AHB_CSW	0x000	W	0x00000002	Control/Status Word, CSW
DAP_AHB_TAR	0x004	W	0x00000000	Transfer Address, TAR
DAP_AHB_DRW	0x00c	W	NA	Data Read/Write, DRW
DAP_AHB_BD0	0x010	W	NA	Bank Data 0, BD0
DAP_AHB_BD1	0x014	W	NA	Bank Data 1, BD1
DAP_AHB_BD2	0x018	W	NA	Bank Data 2, BD2
DAP_AHB_BD3	0x01c	W	NA	Bank Data 3, BD3
DAP_DEBUG_ROM	0xf8	W	NA	Debug ROM table
DAP_AHB_IDR	0xfc	W	0x14770002	Identification Register, IDR

**5.4.4 DAP AHB-AP Detailed Register Description****DAP\_AHB\_CSW**

Address: AHBAP\_BASE + offset(0x000)

Control/Status Word

Bits	Attr	Reset Value	Description
31	-	-	Reserved
30	RW	0x0	<p>Specifies that a secure transfer is requested. SProt HIGH indicates a non-secure transfer. SProt LOW indicates a secure transfer.</p> <ul style="list-style-type: none"> <li>If this bit is LOW, and SPIDEN is HIGH, HPROT[6] is asserted LOW on an AHB transfer.</li> <li>If this bit is LOW, and SPIDEN is LOW, HPROT[6] is asserted HIGH and the AHB transfer is not initiated.</li> <li>If this bit is HIGH, the state of SPIDEN is ignored. HPROT[6] is HIGH.</li> </ul> <p>Reset value: 1'b1, Non-secure</p>
29	-	-	Reserved
28:24	RW	0x0	Specifies the protection signal encoding to be output on HPROT[4:0].
23	RO	0x0	Indicates the status of the SPIDEN port. If SPIStatus is LOW, no secure AHB transfers are

			carried out.
22:12	-	-	Reserved
11:8	RW	0x0	Specifies the mode of operation. 4'b0000: Normal download/upload model 4'b0001-4'b1111: Reserved
7	RO	0x0	Transfer in progress. This field indicates if a transfer is currently in progress on the AHB master port
6	RO	0x0	Indicates the status of the DBGEN port. If DbgStatus is LOW, no AHB transfers are carried out. 1'b1: AHB transfers permitted. 1'b0: AHB transfers not permitted.
5:4	RW	0x0	Auto address increment and packing mode on Read or Write data access. Only increments if the current transaction completes without an Error response and the transaction is not aborted. Auto address incrementing and packed transfers are not performed on access to Banked Data registers 0x10-0x1C. The status of these bits is ignored in these cases. Increments and wraps within a 1KB address boundary, for example, for word incrementing from 0x1400-0x17FC. If the start is at 0x14A0, then the counter increments to 0x17FC, wraps to 0x1400, then continues incrementing to 0x149C. 2'b00: Auto increment OFF. 2'b01: Increment, single. Single transfer from corresponding byte lane. 2'b10: Increment, packed Word: Same effect as single increment. Byte/Halfword: Packs four 8-bit transfers or two 16-bit transfers into a 32-bit DAP transfer. Multiple transactions are carried out on the AHB interface. 2'b11: Reserved SBZ, no transfer. Size of address increment is defined by the Size field, bits [2:0].
3	-	-	Reserved
2:0	RW	0x2	Size of the data access to perform: 3'b000: 8 bits 3'b001: 16 bits 3'b010: 32 bits 3'b011-3'b111: Reserved

**DAP\_AHB\_TAR**

Address: AHBAP\_BASE + offset(0x004)

Control/Status Word

Bits	Attr	Reset Value	Description
31:0	RW	0x0	Address of the current transfer.

**DAP\_AHB\_DRW**

Address: AHBAP\_BASE + offset(0x00c)

Control/Status Word

Bits	Attr	Reset Value	Description
31:0	RW	0x0	Write mode: Data value to write for the current

			transfer. Read mode: Data value read from the current transfer.
--	--	--	--

**DAP\_AHB\_BD0- DAP\_AHB\_BD3**

Address: APBAP\_BASE + offset(0x010) - APBAP\_BASE + offset(0x01c)

Bank Data 0-3

Bits	Attr	Reset Value	Description
31:0	RW	0x0	If DAPADDR[7:4] = 0x0001, so accessing AHB-AP registers in the range 0x10-0x1C, then the derived HADDR[31:0] is: <ul style="list-style-type: none"> <li>• Write mode: Data value to write for the current transfer to external address TAR[31:4]+ DAPADDR[3:2] + 2'b00.</li> <li>• Read mode: Data value read from the current transfer from external address TAR[31:4]+ DAPADDR[3:2] + 2'b00.</li> </ul> Auto address incrementing is not performed on DAP accesses to BD0-BD3. Banked transfers are only supported for word transfers. Non-word banked transfers are reserved and unpredictable. Transfer size is currently ignored for banked transfers.

**DAP\_DEBUG\_ROM**

Address: 0xf8

ROM address

Bits	Attr	Reset Value	Description
31:0	RO	-	Base address of a ROM table. The ROM provides a look-up table for system components.

**DAP\_AHB\_IDR**

Address: APBAP\_BASE + offset(0x0fc)

Bits	Attr	Reset Value	Description
31:28	R	0x4	Revision. Reset value is 0x4 for AHB-AP.
27:24	R	0x4	JEDEC bank. 0x4 indicates ARM Limited.
23:17	R	0x3b	JEDEC code. 0x3B indicates ARM Limited.
16	R	0x1	Memory AP. 0x1 indicates a standard register map is used.
15:8	R	0x00	Reserved
7:0	R	0x01	Identity value. Reset value is 0x01 for AHB-AP.

**5.4.5 DAP-ROM register summary**

Name	Offset	Size	Reset Value	Description
DAP_ROMENTRY0	0x0000	W	0x00001003	CTI4 entry register
DAP_ROMENTRY1	0x0004	W	0x00002003	TPIU entry register
DAP_ROMENTRY2	0x0008	W	0x00003003	Trace Funnel register
DAP_ROMENTRY3	0x000c	W	0x00004003	Cortex-A9 ROM entry register
DAP_ROM_PERIPHID4	0x0fd0	W	0x00000004	Peripheral ID4

DAP_ROM_PERIPHID5	0x0fd4	W	0x00000000	Peripheral ID5
DAP_ROM_PERIPHID6	0x0fd8	W	0x00000000	Peripheral ID6
DAP_ROM_PERIPHID7	0x0fdc	W	0x00000000	Peripheral ID7
DAP_ROM_PERIPHID0	0x0fe0	W	0x000000c4	Peripheral ID0
DAP_ROM_PERIPHID1	0x0fe4	W	0x000000b4	Peripheral ID1
DAP_ROM_PERIPHID2	0x0fe8	W	0x0000006b	Peripheral ID2
DAP_ROM_PERIPHID3	0x0fec	W	0x00000020	Peripheral ID3
DAP_ROM_COMPONID0	0x0ff0	W	0x0000000d	Component ID0
DAP_ROM_COMPONID1	0x0ff4	W	0x00000010	Component ID1
DAP_ROM_COMPONID2	0x0ff8	W	0x00000005	Component ID2
DAP_ROM_COMPONID3	0x0ffc	W	0x000000b1	Component ID3

#### 5.4.6 DAP-ROM Detailed Register Description

##### DAP\_ROMENTRY0

Address: DAPROM\_BASE + offset(0x0000)

TPIU entry register

Bits	Attr	Reset Value	Description
31:0	R	0x00001003	TPIU entry register

##### DAP\_ROMENTRY1

Address: DAPROM\_BASE + offset(0x0004)

Cortex-A9 Debug entry register

Bits	Attr	Reset Value	Description
31:0	R	0x00002003	Cortex-A9 Debug entry register

##### DAP\_ROMENTRY2

Address: DAPROM\_BASE + offset(0x0008)

Cortex-A9 ETM entry register

Bits	Attr	Reset Value	Description
31:0	R	0x00003003	Cortex-A9 ETM entry register

##### DAP\_ROMENTRY3

Address: DAPROM\_BASE + offset(0x000c)

Cortex-A9 CTI entry register

Bits	Attr	Reset Value	Description
31:0	R	0x00004003	Cortex-A9 CTI entry register

##### DAP\_ROM\_PERIPHID4

Address: DAPROM\_BASE + offset(0x0fd0)

Peripheral ID4

Bits	Attr	Reset Value	Description
31:0	R	0x00000004	Peripheral ID4

##### DAP\_ROM\_PERIPHIDS

Address: DAPROM\_BASE + offset(0x0fd4)

Peripheral ID5

Bits	Attr	Reset Value	Description
31:0	R	0x00000000	Peripheral ID5

##### DAP\_ROM\_PERIPHID6

Address: DAPROM\_BASE + offset(0x0fd8)

Peripheral ID6

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x00000000	Peripheral ID6

**DAP\_ROM\_PERIPHID7**

Address: DAPROM\_BASE + offset(0x0fdc)

Peripheral ID7

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x00000000	Peripheral ID7

**DAP\_ROM\_PERIPHID0**

Address: DAPROM\_BASE + offset(0x0fe0)

Peripheral ID0

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x000000c4	Peripheral ID0

**DAP\_ROM\_PERIPHID1**

Address: DAPROM\_BASE + offset(0x0fe4)

Peripheral ID1

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x000000b4	Peripheral ID1

**DAP\_ROM\_PERIPHID2**

Address: DAPROM\_BASE + offset(0x0fe8)

Peripheral ID2

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x00000006b	Peripheral ID2

**DAP\_ROM\_PERIPHID3**

Address: DAPROM\_BASE + offset(0x0fec)

Peripheral ID3

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x000000020	Peripheral ID3

**DAP\_ROM\_COMPOnid0**

Address: DAPROM\_BASE + offset(0x0ff0)

Component ID0

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x00000000d	Component ID0

**DAP\_ROM\_COMPOnid1**

Address: DAPROM\_BASE + offset(0x0ff4)

Component ID0

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x000000010	Component ID1

**DAP\_ROM\_COMPOnid2**

Address: DAPROM\_BASE + offset(0x0ff8)

Component ID0

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x000000005	Component ID2

**DAP\_ROM\_COMPONENTID3**

Address: DAPROM\_BASE + offset(0x0ffC)

Component ID0

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x000000b1	Component ID3

**5.4.7 PTM register summary**

<b>Name</b>	<b>Offset</b>	<b>Size</b>	<b>Reset Value</b>	<b>Description</b>
PTM_ETMCR	0x0	W	0x401	Main control
PTM_ETMCCR	0x4	W	0x8d294004	Configuration code
PTM_ETMTE	0x8	W	0x0	Trigger event
PTM_ETMSR	0x10	W	0x0	Status
PTM_ETMSCR	0x14	W	0x0	System configuration
PTM_ETMTSSCR	0x18	W	0x0	TraceEnable Start/Stop control
PTM_ETMTEE	0x20	W	0x0	TraceEnable event
PTM_ETMTECR1	0x24	W	0x0	TraceEnable Control
PTM_ETMACVR1-8	0x40-0x5c	W	0x0	Address comparator value
PTM_ETMACTR1-8	0x80-0x9c	W	0x0	Address comparator access type
PTM_ETMCNTRLDV R1-2	0x140-0x144	W	0x0	Counter load value
PTM_ETMCNTENR1 -2	0x150-0x154	W	0x0	Counter enable
PTM_ETMCNTRLDE VR1-2	0x160-0x164	W	0x0	Counter reload event
PTM_ETMCNTVR1-2	0x170-0x174	W	0x0	Counter value
PTM_SSTE1-SSTE6	0x180-0x194	W	0x0	Sequencer status transition event
PTM_CSS	0x19c	W	0x0	Current sequencer state
PTM_EOE1-PTM_EOE2	0x1a0-0x1a4	W	0x0	External output event
PTM_CICV1	0x1b0	W	0x0	Context ID comparator value
PTM_CICM	0x1bc	W	0x0	Context ID comparator mask
PTM_ETMSYNCFR	0x1e0	W	0x0	Synchronization frequency
PTM_ETMIDR	0x1e4	W	0x411cf301	ID register
PTM_ETMCCR	0x1e8	W	0x00c019a2	Configuration code extension
PTM_ETMEXTINSEL R	0x1ec	W	0x0	Extended external input selection
PTM_TE	0x1f8	W	0x0	Timestamp Event
PTM_ETMAUXCR	0x1fc	W	0x0	Auxiliary control register
PTM_ETMTRACEID R	0x200	W	0x0	CoreSight trace ID
PTM_OSLSR	0x304	W	0x0	OS lock status
PTM_ETMPDSR	0x314	W	0x1	Device power-down status
PTM_ITMISCOUT	0xedc	W	0x0	Miscellaneous outputs
PTM_ITMISCIN	0xee0	W	0x0	Miscellaneous inputs
PTM_ITTRIGGER	0xee8	W	0x0	Trigger register
PTM_ITATBDA0	0xec	W	0x0	ATB data 0

PTM_ITATBCTR2	0xef0	W	0x0	ATB control 2
PTM_ITATBID	0xef4	W	0x0	ATB identification
PTM_ITATBCTR0	0xef8	W	0x0	ATB control 0
PTM_ETMITCTRL	0xf00	W	0x0	Integration mode control
PTM_AS	0xfb8	W	0x0	Authentication status
PTM_DC	0xfc8	W	0x0	Device configuration
PTM_DT	0xfcfc	W	0x0	Device type
PTM_PID4	0xfd0	W	0x4	Peripheral ID4
PTM_PID5	0xfd4	W	0x0	Peripheral ID5
PTM_PID6	0xfd8	W	0x0	Peripheral ID6
PTM_PID7	0fdc	W	0x0	Peripheral ID7
PTM_PID0	0xfe0	W	0x50	Peripheral ID0
PTM_PID1	0xfe4	W	0xb9	Peripheral ID1
PTM_PID2	0xfe8	W	0x1b	Peripheral ID2
PTM_PID3	0fec	W	0x0	Peripheral ID3
PTM_CID0	0xff0	W	0xd	Component ID0
PTM_CID1	0xff4	W	0x90	Component ID1
PTM_CID2	0xff8	W	0x5	Component ID2
PTM_CID3	0ffc	W	0xb1	Component ID3

#### 5.4.8 PTM Detailed Register Description

##### PTM\_ETMCR

Address: PTM\_BASE + offset(0x000)

Main Control Register

Bits	Attr	Reset Value	Description
31:30	-	-	Reserved
29	RW	0x0	Return stack enable 1'b0: disabled 1'b1: enabled
28	RW	0x0	Timestamp enable 1'b0: disabled 1'b1: enabled
27:25	RW	0x0	Processor select
24	R	0x0	Reserved
23:16	-	-	Reserved
15:14	RW	0x0	ContextIDSize 2'b00: no context ID tracing 2'b01: context ID bits [7:0] traced 2'b10: context ID bits [15:0] traced 2'b11: context ID bits [31:0] traced. On reset, this bit is set to b00, no context ID tracing
13	-	-	Reserved
12	RW	0x0	CycleAccurate 1'b0: cycle counting disabled 1'b1: cycle counting enabled On reset this bit is set to 1'b0, no cycle counting.
11	-	-	Reserved
10	RW	0x1	Programming Bit This bit must be set to 1'b1 when the PTM is being programmed. On a PTM reset this bit is set to 1'b1.

9	RW	0x0	Debug request control When set to 1'b1 and the trigger event occurs, the PTMDBGRQ output is asserted until PTMDBGACK is observed. This enables a debugger to force the processor into Debug state. On PTM reset this bit is set to 1'b0.
8	RW	0x0	Branch Output When this bit is set to 1'b1, addresses are output for all executed branches, both direct and indirect. On PTM reset this bit is set to 1'b0.
7	R	0x0	Stall processor
6:1	-	-	Reserved
0	RW	0x1	PowerDown This bit enables external control of the PTM. This bit must be cleared by the trace software tools at the beginning of a debug session. When this bit is set to 1'b0, both the PTM and the trace interface in the processor are enabled. To avoid corruption of trace data, this bit must not be set before the Programming Status bit in the PTM Status Register has been read as 1. On PTM reset this bit is set to 1'b1.

**PTM\_ETMCCR**

Address: PTM\_BASE + offset(0x004)

Configuration Code Register

Bits	Attr	Reset Value	Description
31	RO	0x1	ID Register present Indicates that the ID Register is present.
30:28	-	-	Reserved
27	RO	0x1	Software access Indicates that software access is supported
26	RO	0x1	Trace stop/start block Indicates that the trace start/stop block is present.
25:24	RO	0x1	Number of Context ID comparators Specifies the number of Context ID comparators, one.
23	RO	0x0	FIFOFULL logic Indicates that it is not possible to stall the processor to prevent FIFO overflow
22:20	RO	0x2	Number of external outputs Specifies the number of external outputs, two.
19:17	RO	0x4	Number of external inputs Specifies the number of external inputs, four.
16	RO	0x1	Sequencer Indicates that the sequencer is present
15:13	RO	0x2	Number of counters Specifies the number of counters, two.
12:4	-	-	Reserved
3:0	RO	0x4	Number of pairs of address comparators Specifies the number of address comparator pairs, four.

**PTM\_ETMSCR**

Address: PTM\_BASE + offset(0x014)

System Configuration Register

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	-	-	Reserved
14:12	RO	-	Number of supported processors minus 1. The value of this field is set by the MAXCORES[2:0] input to the PTM
11:9	-	-	Reserved
8	RO	-	Read Only, as 1'b0 - FIFOFULL is not supported.
7:0	-	-	Reserved

**PTM\_ETMTSSCR**

Address: PTM\_BASE + offset(0x018)

TraceEnable Start/Stop Control Register

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	-	-	Reserved
23:16	RW	0x0	When a bit is set to 1, it selects a single address comparator (8-1) as a stop address for the TraceEnable Start/Stop block. For example, if you set bit [16] to 1 it selects single address comparator 1 as a stop address.
15:8	-	-	Reserved
7:0	RW	0x0	When a bit is set to 1, it selects a single address comparator (8-1) as a start address for the TraceEnable Start/Stop block. For example, if you set bit [0] to 1 it selects single address comparator 1 as a start address.

**PTM\_ETMTECR1**

Address: PTM\_BASE + offset(0x024)

TraceEnable Control Register1

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	-	-	Reserved
25	RW	0x0	Trace start/stop control enable. The possible values of this bit are: 1'b0: Tracing is unaffected by the trace start/stop logic. 1'b1: Tracing is controlled by the trace on and off addresses configured for the trace start/stop logic. The trace start/stop resource is not affected by the value of this bit.
24	RW	0x0	Exclude/include flag. The possible values of this bit are: 1'b0: Include. The specified address range comparators indicate the regions where tracing can occur. No tracing occurs outside this region. 1'b1: Exclude. The specified address range comparators indicate regions to be excluded from the trace. When outside an exclude region,

			tracing can occur.
23:4	-	-	Reserved
3:0	RW	0x0	When a bit is set to 1, it selects an address range comparator, 4-1, for include/exclude control. For example, bit [0] set to 1 selects address range comparator 1

**PTM\_ETMACVR1-8**

Address: PTM\_BASE + offset(0x040-0x05c)

ETMACVR1-8

Bits	Attr	Reset Value	Description
31:0	RW	0x0	Address comparator value register

**PTM\_ETMACTR1-8**

Address: PTM\_BASE + offset(0x080-0x09c)

ETMACTR1-8

Bits	Attr	Reset Value	Description
31:0	RW	0x0	Address comparator access type register

**PTM\_ETMCNTRLDVR1-2**

Address: PTM\_BASE + offset(0x140-0x144)

ETMCNTRLDVR1-2

Bits	Attr	Reset Value	Description
31:0	RW	0x0	Reload value

**PTM\_ETMCNTENR1-2**

Address: PTM\_BASE + offset(0x150-0x154)

ETMCNTENR1-2

Bits	Attr	Reset Value	Description
31:0	RW	0x0	Enable Event

**PTM\_ETMCNTRLDEVR1-2**

Address: PTM\_BASE + offset(0x160-0x164)

ETMCNTRLDEVR1-2

Bits	Attr	Reset Value	Description
31:0	RW	0x0	Reload Event

**PTM\_ETMCNTVR1-2**

Address: PTM\_BASE + offset(0x160-0x164)

ETMCNTVR1-2

Bits	Attr	Reset Value	Description
31:0	RW	0x0	Value

**PTM\_ETMSYNCFR**

Address: PTM\_BASE + offset(0x1e0)

Bits	Attr	Reset Value	Description
31:0	RW	0x0	The ETMSYNCFR holds the trace synchronization frequency value. Bits [2:0] of this register are not implemented and read as zero (RAZ).

**PTM\_ETMIDR**

Address: PTM\_BASE + offset(0x1e4)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x41	Implementor code. This field reads as 0x41, ASCII code for A, indicating ARM Limited.
23:20	-	-	Reserved
19	RO	0x1	Support for Security Extensions. The value of this bit is 1, indicating that the processor implements the ARM architecture Security Extensions.
18	RO	0x1	Support for 32-bit Thumb instructions. The value of this bit is 1, indicating that a 32-bit Thumb instruction is traced as a single instruction.
17:12	-	-	Reserved
11:8	RO	0x3	Major architecture version number.
7:4	RO	0x0	Major architecture version number.
3:0	RO	-	Implementation revision

**PTM\_ETMCER**

Address: PTM\_BASE + offset(0x1e8)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RO	0x0	Reserved
25	RO	0x0	Timestamps not generated for DMB/DSB.
24	RO	0x0	MB/DSB instructions are not treated as waypoints.
23	RO	0x1	Return stack implemented.
22	RO	0x1	Timestamping implemented.
21:16	-	-	Reserved
15:13	RO	0x0	Specifies the number of instrumentation resources.
12	-	-	Reserved
11	RO	0x1	b1 - Indicates that all registers, except some Integration Test Registers, are readable.
10:3	RO	0x34	Specifies the size of the extended external input bus, 52.
2:0	RO	0x2	Specifies the number of extended external input selectors, 2.

**PTM\_ETMEXTINSELR**

Address: PTM\_BASE + offset(0x1ec)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	-	-	Reserved
13:8	RW	0x0	Second extended external input selector
7:6	-	-	Reserved
5:0	RW	0x0	First extended external input selector

**PTM\_ETMAUXCR**

Address: PTM\_BASE + offset(0x1fc)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	-	-	Reserved
3	RW	0x0	Force insertion of synchronization packets, regardless of current trace activity.

			Possible values for this bit are: 1'b0: Synchronization packets delayed when trace activity is high. This is the reset value. 1'b1: Synchronization packets inserted regardless of trace activity. This bit might be set if synchronization packets occur too far apart. Setting this bit might cause the trace FIFO to overflow more frequently when trace activity is high.
2	RW	0x0	Specifies whether the PTM issues waypoint update packets if there are more than 4096 bytes between waypoints. Possible values for this bit are: 1'b0: PTM always issues update packets if there are more than 4096 bytes between waypoints. This is the reset value. 1'b1: PTM does not issue waypoint update packets unless required to do so as the result of an exception or debug entry.
1	RW	0x0	Specifies whether the PTM issues a timestamp on a barrier instruction. Possible values for this bit are: 1'b0: PTM issues timestamps on barrier instructions. This is the reset value. 1'b1: PTM does not issue timestamps on barriers.
0	RW	0x0	Specifies whether the PTM enters overflow state when synchronization is requested, and the previous synchronization sequence has not yet completed. This does not affect entry to overflow state when the FIFO becomes full. Possible values for this bit are: 1'b0: Forced overflow enabled. This is the reset value. 1'b1: Forced overflow disabled.

**PTM\_ETMTRACEIDR**

Address: PTM\_BASE + offset(0x200)

Bits	Attr	Reset Value	Description
31:7	-	-	Reserved
6:0	RW	0x0	Before trace is generated, you must program this register with a non-reserved value. Reserved values are 0x00 and any value in the range 0x70-0x7F. The reset value of this register is 0x00

**PTM\_ETMPDSR**

Address: PTM\_BASE + offset(0x314)

Bits	Attr	Reset Value	Description
31:0	RO	0x1	This register always reads as 0x00000001, indicating that the PTM Trace Registers can be accessed.

**PTM\_OSLSR**

Address: PTM\_BASE + offset(0x304)

Bits	Attr	Reset Value	Description

31:0	RO	0x0	For the PTM, the OSLSR Reads As Zero (RAZ) to show that OS Locking is not implemented.
------	----	-----	--

**PTM\_ITMISCONT**

Address: PTM\_BASE + offset(0xedc)

Bits	Attr	Reset Value	Description
31:10	-	-	Reserved
9:8	WO	0x0	Drives the PTMEXTOUT[1:0] outputs
7:6	-	-	Reserved
5	WO	0x0	Drives the PTMIDLEnACK output
4	WO	0x0	Drives the PTMDBGREQ output
3:0	WO	0x0	Reserved

**PTM\_ITMISCIN**

Address: PTM\_BASE + offset(0xee0)

Bits	Attr	Reset Value	Description
31:7	-	-	Reserved
6	RO	0x0	Returns the value of the STANDBYWFI input
5	-	-	Reserved
4	RO	0x0	Returns the value of the PTMDBGACK input
3:0	RO	0x0	Returns the value of the EXTIN[3:0] inputs

**PTM\_ITTRIGGER**

Address: PTM\_BASE + offset(0xee8)

Bits	Attr	Reset Value	Description
31:1	-	-	Reserved
0	WO	0x0	Drives the PTMTRIGGER output

**PTM\_ITATBDA0**

Address: PTM\_BASE + offset(0xeee)

Bits	Attr	Reset Value	Description
31:5	-	-	Reserved
4	WO	-	Drives the ATDATAM[31] output
3	WO	-	Drives the ATDATAM[23] output
2	WO	-	Drives the ATDATAM[15] output
1	WO	-	Drives the ATDATAM[7] output
0	WO	-	Drives the ATDATAM[0] output

**PTM\_ITATBCTR2**

Address: PTM\_BASE + offset(0xef0)

Bits	Attr	Reset Value	Description
31:2	-	-	Reserved
1	RO	-	Returns the value of the AFVALIDM input
0	RO	-	Returns the value of the ATREADYM input

**PTM\_ITATBID**

Address: PTM\_BASE + offset(0xef4)

Bits	Attr	Reset Value	Description
31:7	-	-	Reserved

6:0	WO	-	Drives the ATIDM[6:0] outputs
-----	----	---	-------------------------------

**PTM\_ITATBCTR0**

Address: PTM\_BASE + offset(0xef8)

Bits	Attr	Reset Value	Description
31:10	-	-	Reserved
9:8	WO	-	Drives the ATBYTESM outputs
7:2	-	-	Reserved
1	WO	-	Drives the AFREADYM output
0	WO	-	Drives the ATVALIDM output

**PTM\_ETMITCTRL**

Address: PTM\_BASE + offset(0xff0)

Bits	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	<p>When bit [0] is set to 1, the PTM enters an integration mode. On reset this bit is cleared to 0.</p> <p>Before entering integration mode, the PTM must be powered up and in programming mode. This means bit [0] of the Main Control Register is set to 0, and bit [10] of the Main Control Register is set to 1.</p> <p>After leaving integration mode, the PTM must be reset before attempting to perform tracing.</p>

**PTM\_PID4**

Address: PTM\_BASE + offset(0xfd0)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:4	RO	0x04	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**PTM\_PID5**

Address: PTM\_BASE + offset(0xfd4)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:4	RO	0x00	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**PTM\_PID6**

Address: PTM\_BASE + offset(0xfd8)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:4	RO	0x00	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**PTM\_PID7**

Address: PTM\_BASE + offset(0xfd0c)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x00	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**PTM\_PID0**

Address: PTM\_BASE + offset(0xfe0)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x50	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**PTM\_PID1**

Address: PTM\_BASE + offset(0xfe4)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0xb9	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**PTM\_PID2**

Address: PTM\_BASE + offset(0xfe8)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x1b	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**PTM\_PID3**

Address: PTM\_BASE + offset(0xfc0)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x00	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**PTM\_CID0**

Address: PTM\_BASE + offset(0xff0)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x0d	The component identification registers identify the PTM as a CoreSight component

**PTM\_CID1**

Address: PTM\_BASE + offset(0xff4)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x90	The component identification registers identify the PTM as a CoreSight component

**PTM\_CID2**

Address: PTM\_BASE + offset(0xff8)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x05	The component identification registers identify the PTM as a CoreSight component

**PTM\_CID3**

Address: PTM\_BASE + offset(0xffc)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0xb1	The component identification registers identify the PTM as a CoreSight component

**5.4.9 Funnel register summary**

<b>Name</b>	<b>Offset</b>	<b>Size</b>	<b>Reset Value</b>	<b>Description</b>
FUNNEL_FCR	0x0	W	0x300	CSTF Control Register
FUNNEL_PCR	0x4	W	0xfac688	CSTF Priority Control Register
FUNNEL_ITATBDATA0	0xeeec	W	0x0	CSTF Integration Test Registers
FUNNEL_ITATBCTR2	0xef0	W	0x0	CSTF Integration Test Registers
FUNNEL_ITATBCTR1	0xef4	W	0x0	CSTF Integration Test Registers
FUNNEL_ITATBCTR0	0xef8	W	0x0	CSTF Integration Test Registers
FUNNEL_IMCR	0xff0	W	0x0	Integration Mode Control Register
FUNNEL_CTSR	0xfa0	W	0xf	Claim Tag Set Register
FUNNEL_CTCR	0xfa4	W	0x0	Claim Tag Clear Register
FUNNEL_LA	0xfb0	W	-	Lock Access
FUNNEL_LS	0xfb4	W	0x0	Lock Status
FUNNEL_AS	0xfb8	W	0x0	Authentication status
FUNNEL_DI	0xfc8	W	0x28	Device ID
FUNNEL_DTI	0xfc	W	0x12	Device Type Identifier
FUNNEL_PID4	0xfd0	W	0x04	Peripheral ID4
FUNNEL_PID0	0xfe0	W	0x08	Peripheral ID0
FUNNEL_PID1	0xfe4	W	0xb9	Peripheral ID1
FUNNEL_PID2	0xfe8	W	0x1b	Peripheral ID2
FUNNEL_PID3	0fec	W	0x00	Peripheral ID3
FUNNEL_CID0	0xff0	W	0xd	Component ID0
FUNNEL_CID1	0xff4	W	0x90	Component ID1
FUNNEL_CID2	0xff8	W	0x05	Component ID2
FUNNEL_CID3	0ffc	W	0xb1	Component ID3

### 5.4.10 Funnel register details

#### FUNNEL\_CR

Address: FUNNEL\_BASE + offset(0x000)

Bits	Attr	Reset Value	Description
31:12	-	-	Reserved
11:8	RW	0x3	Minimum hold time[3:0] The formatting scheme can easily become inefficient if fast switching occurs, so, where possible, this must be minimized. If a source has nothing to transmit, then another source is selected irrespective of the minimum number of cycles. Reset is 0x3. The CSTF holds for the minimum hold time and one additional cycle. The maximum value that can be entered is 0xE and this equates to 15 cycles. 0xF is reserved.
7	RW	0x0	Enable Slave port 7 Setting this bit enables this input, or slave, port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme. The reset value is all clear, that is, all ports disabled.
6	RW	0x0	Enable Slave port 6 Setting this bit enables this input, or slave, port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme. The reset value is all clear, that is, all ports disabled.
5	RW	0x0	Enable Slave port 5 Setting this bit enables this input, or slave, port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme. The reset value is all clear, that is, all ports disabled.
4	RW	0x0	Enable Slave port 4 Setting this bit enables this input, or slave, port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme. The reset value is all clear, that is, all ports disabled.
3	RW	0x0	Enable Slave port 3 Setting this bit enables this input, or slave, port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme. The reset value is all clear, that is, all ports disabled.
2	RW	0x0	Enable Slave port 2 Setting this bit enables this input, or slave, port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme. The reset value is all clear, that is, all ports disabled.
1	RW	0x0	Enable Slave port 1 Setting this bit enables this input, or slave, port. If the bit is not set then this has the effect of

			excluding the port from the priority selection scheme. The reset value is all clear, that is, all ports disabled.
0	RW	0x0	Enable Slave port 0 Setting this bit enables this input, or slave, port. If the bit is not set then this has the effect of excluding the port from the priority selection scheme. The reset value is all clear, that is, all ports disabled.

**FUNNEL\_PCR**

Address: FUNNEL\_BASE + offset(0x004)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	-	-	Reserved
23:21	RW	0x0	PriPort 7 Priority value of the eighth port. The value written into this location is the value that you want to assign the eighth slave port.
20:18	RW	0x0	PriPort 6 7th port priority value.
17:15	RW	0x0	PriPort 5 6th port priority value.
14:12	RW	0x0	PriPort 4 5th port priority value.
11:9	RW	0x0	PriPort 3 4th port priority value.
8:6	RW	0x0	PriPort 2 3th port priority value.
5:3	RW	0x0	PriPort 1 2th port priority value.
2:0	RW	0x0	PriPort 0 Priority value of the first slave port. The value written into this location is the value that you want to assign the first slave port

**FUNNEL\_ITATBDATA0**

Address: FUNNEL\_BASE + offset(0xeeec)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	-	-	Reserved
4	RW	0x0	the value of ATDATAS<31>
3	RW	0x0	the value of ATDATAS<23>
2	RW	0x0	the value of ATDATAS<15>
1	RW	0x0	the value of ATDATAS<7>
0	RW	0x0	the value of ATDATAS<0>

**FUNNEL\_ITATBCTR2**

Address: FUNNEL\_BASE + offset(0xef0)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	-	-	Reserved
1	RW	0x0	the value of AFVALIDM
0	RW	0x0	the value of ATREADYM

**FUNNEL\_ITATBCTR1**

Address: FUNNEL\_BASE + offset(0xef4)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	-	-	Reserved
6:0	RW	0x0	the value of ATIDS

**FUNNEL\_ITATBCTRO**

Address: FUNNEL\_BASE + offset(0xef4)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	-	-	Reserved
9:8	RW	0x0	the value of ATBYTESS<n>
7:2	-	-	Reserved
1	RW	0x0	the value of AFREADYS<n>
0	RW	0x0	Read the value of ATVALIDS<n>

**FUNNEL\_CTS- FUNNEL\_CTC**

Address: FUNNEL\_BASE + offset(0xfa0-0xfa4)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	-	-	Reserved
3:0	RW	0x0	The CSTF implements a four-bit claim tag. The use of bits [3:0] is software defined.

**FUNNEL\_LA- FUNNEL\_LS**

Address: FUNNEL\_BASE + offset(0xfb0-0xfb4)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	-	-	Reserved
2:0	RW	0x3	The CSTF implements two memory maps controlled through PADDRDBG31. When PADDRDBG31 is HIGH, the Lock Status Register reads as 0x0 indicating that no lock exists. When PADDRDBG31 is LOW, the Lock Status Register reads as 0x3 from reset. This indicates a 32-bit lock access mechanism is present and is locked.

**FUNNEL\_AS**

Address: FUNNEL\_BASE + offset(0xfb8)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:0	RW	0x0	Reports the required security level. This is set to 0x00 for functionality not implemented.

**FUNNEL\_DID**

Address: FUNNEL\_BASE + offset(0xfc8)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RW	0x0	The CSTF implements a static priority scheme.
3:0	RW	0x8	This is the value of the Verilog define PORTCOUNT and represents the number of input ports connected. By default all 8 ports are connected. 0x0 and 0x1 are illegal values.

**FUNNEL\_DTID**

Address: FUNNEL\_BASE + offset(0xfc0)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:0	RW	0x12	A value of 0x12 identifies this device as a trace link (0x2) and specifically as a funnel/router (0x1)

**FUNNEL\_PID4**

Address: FUNNEL\_BASE + offset(0xfd0)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x04	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**FUNNEL\_PID5**

Address: FUNNEL\_BASE + offset(0xfd4)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x00	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**FUNNEL\_PID6**

Address: FUNNEL\_BASE + offset(0xfd8)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x00	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**FUNNEL\_PID7**

Address: FUNNEL\_BASE + offset(0fdc)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x00	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**FUNNEL\_PID0**

Address: FUNNEL\_BASE + offset(0xfe0)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x50	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**FUNNEL\_PID1**

Address: FUNNEL\_BASE + offset(0xfe4)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0xb9	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**FUNNEL\_PID2**

Address: FUNNEL\_BASE + offset(0xfe8)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x1b	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**FUNNEL\_PID3**

Address: FUNNEL\_BASE + offset(0xfc)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x00	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**FUNNEL\_CID0**

Address: FUNNEL\_BASE + offset(0xff0)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x0d	The component identification registers identify the PTM as a CoreSight component

**FUNNEL\_CID1**

Address: FUNNEL\_BASE + offset(0xff4)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x90	The component identification registers identify the PTM as a CoreSight component

**FUNNEL\_CID2**

Address: FUNNEL\_BASE + offset(0xff8)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x05	The component identification registers identify the PTM as a CoreSight component

**FUNNEL\_CID3**

Address: FUNNEL\_BASE + offset(0ffc)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0xb1	The component identification registers identify the PTM as a CoreSight component

### 5.4.11 CTI register summary

Name	Offset	Size	Reset Value	Description
CTI_CTICONTROL	0x000	W	0x0	CTI Control Register
CTI_CTIINTACK	0x010	W	-	CTI Interrupt Acknowledge Register
CTI_CTIAPPSET	0x014	W	0x0	CTI Application Trigger Set Register
CTI_CTIAPPCLEAR	0x018	W	0x0	CTI Application Trigger Clear Register
CTI_CTIAPPPULSE	0x01c	W	0x0	CTI Application Pulse Register
CTI_CTIINEN	0x020-0x03c	W	0x0	CTI Trigger to Channel Enable Registers, CTIINENO-7
CTI_CTIOUTEN	0x0a0-0x0bc	W	0x0	CTI Channel to Trigger Enable Registers, CTIOUTENO-7
CTI_CТИTRIGINSTATUS	0x130	W	0x0	CTI Trigger In Status Register, CTITRIGINSTATUS
CTI_CТИCHINSTATUS	0x138	W	-	CTI Channel In Status Register, CTICHINSTATUS
CTI_CТИCHOUTSTATUS	0x13c	W	0x0	CTI Channel Out Status Register
CTI_CТИGATE	0x140	W	0xf	Enable CTI Channel Gate Register
CTI_ASICCTL	0x144	W	0x0	External Multiplexor Control Register
CTI_ITCHINACK	0xedc	W	0x0	ITCHINACK Register
CTI_ITTRIGINACK	0xee0	W	0x0	ITTRIGINACK Register
CTI_ITCHOUT	0xee4	W	0x0	ITCHOUT Register
CTI_ITTRIGOUT	0xee8	W	0x0	ITTRIGOUT Register
CTI_ITCHOUTACK	0xec	W	0x0	ITCHOUTACK Register
CTI_ITTRIGOUTACK	0xef0	W	0x0	ITTRIGOUTACK Register
CTI_ITCHIN	0xef4	W	0x0	ITCHIN Register
CTI_ITTRIGIN	0xef8	W	0x0	ITTRIGIN Register
CTI_ITCTRL	0xf00	W	0x0	ITCTRL Register
CTI_CTSR	0xfa0	W	0xf	Claim Tag Set Register
CTI_CTCR	0xfa4	W	0x0	Claim Tag Clear Register
CTI_LA	0xfb0	W	-	Lock Access
CTI_LS	0xfb4	W	0x0	Lock Status
CTI_AS	0xfb8	W	0x0	Authentication status
CTI_DI	0xfc8	W	0x28	Device ID
CTI_DTI	0xfcc	W	0x12	Device Type Identifier
CTI_PID4	0xfd0	W	0x04	Peripheral ID4
CTI_PID5	0xfd4	W	0x00	Peripheral ID5
CTI_PID6	0xfd8	W	0x00	Peripheral ID6
CTI_PID7	0fdc	W	0x00	Peripheral ID7
CTI_PID0	0xfe0	W	0x08	Peripheral ID0
CTI_PID1	0xfe4	W	0xb9	Peripheral ID1
CTI_PID2	0xfe8	W	0x1b	Peripheral ID2
CTI_PID3	0fec	W	0x00	Peripheral ID3

CTI_CID0	0xff0	W	0x0d	Component ID0
CTI_CID1	0xff4	W	0x90	Component ID1
CTI_CID2	0xff8	W	0x05	Component ID2
CTI_CID3	0ffc	W	0xb1	Component ID3

#### 5.4.12 CTI register details

##### CTI\_CTLCONTROL

Address: CTI\_BASE + offset(0x000)

Bits	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	GLBEN Enables or disables the ECT: 1'b0: disabled (reset) 1'b1: enabled. When disabled, all cross triggering mapping logic functionality is disabled for this processor

##### CTI\_CTLINTACK

Address: CTI\_BASE + offset(0x010)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	WO	-	INTACK Acknowledges the corresponding CTITRIGOUT output: 1'b1: CTITRIGOUT is acknowledged and is cleared when MAPTRIGOUT is LOW. 1'b0: no effect. There is one bit of the register for each CTITRIGOUT output.

##### CTI\_CTLAPPSET

Address: CTI\_BASE + offset(0x014)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RW	0x0	APPSET Setting a bit HIGH generates a channel event for the selected channel. Read: 1'b0: application trigger inactive (reset) 1'b1: application trigger active. Write: 1'b0: no effect 1'b1: generate channel event. There is one bit of the register for each channel.

##### CTI\_CTLAPPCLEAR

Address: CTI\_BASE + offset(0x018)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	WO	-	Clears corresponding bits in the CTIAPPSET register. 1'b1: application trigger disabled in the CTIAPPSET register

			1'b0: no effect There is one bit of the register for each channel.
--	--	--	---

**CTI\_CTIAPPPULSE**

Address: CTI\_BASE + offset(0x01c)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	WO	-	APPULSE Setting a bit HIGH generates a channel event pulse for the selected channel. Write: 1'b1: channel event pulse generated for one CTICLK period 1'b0: no effect There is one bit of the register for each channel.

**CTI\_CTIINENO-7**

Address: CTI\_BASE + offset(0x020-0x03c)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RW	0x0	TRIGINEN Enables a cross trigger event to the corresponding channel when an CTITRIGIN is activated. 1'b1: enables the CTITRIGIN signal to generate an event on the respective channel of the CTM. There is one bit of the register for each of the four channels. For example in register CTIINENO, TRIGINEN[0] set to 1 enables CTITRIGIN onto channel 0. 1'b0: disables the CTITRIGIN signal from generating an event on the respective channel of the CTM.

**CTI\_CTIOUTENO-7**

Address: CTI\_BASE + offset(0x0a0-0x0bc)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RW	0x0	TRIGOUTEN Changing the value of this bit from a 0 to a 1 enables a channel event for the corresponding channel to generate an CTITRIGOUT output: 1'b0: the channel input (CTICHIN) from the CTM is not routed to the CTITRIGOUT output 1'b1: the channel input (CTICHIN) from the CTM is routed to the CTITRIGOUT output. There is one bit for each of the four channels. For example in register CTIOUTENO, enabling bit 0 enables CTICHIN[0] to cause a trigger event on the CTITRIGOUT[0] output.

**CTI\_CТИTRIGINSTATUS**

Address: CTI\_BASE + offset(0x130)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved

7:0	RW	0x0	TRIGINSTATUS Shows the status of the CTITRIGIN inputs: 1'b1: CTITRIGIN is active 1'b0: CTITRIGIN is inactive. Because the register provides a view of the raw CTITRIGIN inputs, the reset value is unknown. There is one bit of the register for each trigger input.
-----	----	-----	---

**CTI\_CTI TRIGOUTSTATUS**

Address: CTI\_BASE + offset(0x134)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x0	TRIGOUTSTATUS Shows the status of the CTITRIGOUT outputs. 1'b1: CTITRIGOUT is active 1'b0: CTITRIGOUT is inactive (reset). There is one bit of the register for each trigger output.

**CTI\_CTI CHINSTATUS**

Address: CTI\_BASE + offset(0x138)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RW	0x0	CTICHINSTATUS Shows the status of the CTICHIN inputs: 1'b1: CTICHIN is active 1'b0: CTICHIN is inactive. Because the register provides a view of the raw CTICHIN inputs from the CTM, the reset value is unknown. There is one bit of the register for each channel input.

**CTI\_CTI CHOUTSTATUS**

Address: CTI\_BASE + offset(0x13c)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RW	0x0	CTICHOUTSTATUS Shows the status of the CTICHOUT outputs. 1'b1: CTICHOUT is active 1'b0: CTICHOUT is inactive (reset). There is one bit of the register for each channel output.

**CTI\_CTI GATE**

Address: CTI\_BASE + offset(0x140)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3	RW	0x0	CTIGATEEN3 Enable CTICHOUT3. Set to 0 to disable channel propagation.
2	RW	0x0	CTIGATEEN2 Enable CTICHOUT2. Set to 0 to disable channel propagation.

1	RW	0x0	CTIGATEEN1 Enable CTICHOUT1. Set to 0 to disable channel propagation.
0	RW	0x0	CTIGATEEN0 Enable CTICHOUT0. Set to 0 to disable channel propagation

**CTI ASICCTL**

Address: CTI\_BASE + offset(0x144)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x0	ASICCTL Implementation-defined ASIC control, value written to the register is output on ASICCTL[7:0]. If external multiplexing of trigger signals is implemented then the number of multiplexed signals on each trigger must be reflected within the Device ID Register. This is done within a Verilog define EXTMUXNUM.

**CTI\_ITCHINACK**

Address: CTI\_BASE + offset(0xedc)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	WO	-	CTCHINACK Set the value of the CTCHINACK outputs

**CTI\_ITTRIGINACK**

Address: CTI\_BASE + offset(0xee0)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	WO	-	CTTRIGINACK Set the value of the CTTRIGINACK outputs

**CTI\_ITCHOUT**

Address: CTI\_BASE + offset(0xee4)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	WO	-	CTCHOUT Set the value of the CTCHOUT outputs

**CTI\_ITTRIGOUT**

Address: CTI\_BASE + offset(0xee8)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	WO	-	CTTRIGOUT Set the value of the CTTRIGOUT outputs

**CTI\_ITCHOUTACK**

Address: CTI\_BASE + offset(0xeec)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RO	0x0	CTCHOUTACK

			Read the values of the CTCHOUTACK inputs
--	--	--	--

**CTI\_ITTRIGOUTACK**

Address: CTI\_BASE + offset(0xef0)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RO	0x0	CTTRIGOUTACK Read the values of the CTTRIGOUTACK inputs

**CTI\_ITCHIN**

Address: CTI\_BASE + offset(0xef4)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RO	0x0	CTCHIN Read the values of the CTCHIN inputs

**CTI\_ITTRIGIN**

Address: CTI\_BASE + offset(0xef8)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RO	0x0	CTTRIGIN Read the values of the CTTRIGIN inputs

**CTI\_CTS- CTI\_CTC**

Address: CTI\_BASE + offset(0xfa0-0xfa4)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RW	0x0	The CTI implements a four-bit claim tag. The use of bits [3:0] is software defined

**CTI\_LA- CTI\_LS**

Address: CTI\_BASE + offset(0xfb0-0xfb4)

Bits	Attr	Reset Value	Description
31:3	-	-	Reserved
2:0	RW	0x3	The CTI implements two memory maps controlled through PADDRDBG31. When PADDRDBG31 is HIGH, the Lock Status Register reads as 0x0 indicating that no lock exists. When PADDRDBG31 is LOW, the Lock Status Register reads as 0x3 from reset. This indicates a 32-bit lock access mechanism is present and is locked.

**CTI\_AS**

Address: CTI\_BASE + offset(0xfb8)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3	RW	0x0	Current value of noninvasive debug enable signals
2	RW	0x0	Non-invasive debug controlled
1	RW	0x0	Current value of invasive debug enable signals
0	RW	0x0	Invasive debug controlled

**CTI\_DID**

Address: CTI\_BASE + offset(0xfc8)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	-	-	Reserved
19:16	RO	0x0	Number of ECT channels available.
15:8	RO	0x0	Number of ECT triggers available.
7:5	-	-	Reserved
4:0	RO	0x0	Indicates the number of multiplexing available on Trigger Inputs and Trigger Outputs using ASICCTL. Default value of 5'b00000 indicating no multiplexing present. Reflects the value of the Verilog `define EXTMUXNUM that you must alter accordingly.

**CTI\_DTID**

Address: CTI\_BASE + offset(0fcc)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:0	RW	0x14	0x14 indicates this device has a major type of debug control logic component (0x4) and sub-type corresponding to cross trigger (0x1).

**CTI\_PID4**

Address: CTI\_BASE + offset(0xfd0)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
3:0	RO	0x3	The CTI is identified as an ARM component with a JEP106 identity at 0x3B and a JEP106 continuation code at 0x4 (fifth bank).

**CTI\_PID0**

Address: CTI\_BASE + offset(0xfe0)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0x0	Middle BCD value of Device number.
3:0	RO	0x6	Lower BCD value of Device number.

**CTI\_PID1**

Address: CTI\_BASE + offset(0xfe4)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	Reserved
7:4	RO	0xb	The CTI is identified as an ARM component with a JEP106 identity at 0x3B and a JEP106 continuation code at 0x4 (fifth bank).
3:0	RO	0x9	Upper BCD value of Device number.

**CTI\_PID2**

Address: CTI\_BASE + offset(0xfe8)

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	-	-	Reserved
2:0	RO	0x4	The CTI is identified as an ARM component with

			a JEP106 identity at 0x3B and a JEP106 continuation code at 0x4 (fifth bank).
3:0	-	-	Reserved

**CTI\_PID3**

Address: CTI\_BASE + offset(0xfec)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:4	RO	0x00	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**CTI\_CID0**

Address: CTI\_BASE + offset(0xff0)

Bits	Attr	Reset Value	Description
31:0	RO	0x0d	The component identification registers identify the PTM as a CoreSight component

**CTI\_CID1**

Address: CTI\_BASE + offset(0xff4)

Bits	Attr	Reset Value	Description
31:0	RO	0x90	The component identification registers identify the PTM as a CoreSight component

**CTI\_CID2**

Address: CTI\_BASE + offset(0xff8)

Bits	Attr	Reset Value	Description
31:0	RO	0x05	The component identification registers identify the PTM as a CoreSight component

**CTI\_CID3**

Address: CTI\_BASE + offset(0ffc)

Bits	Attr	Reset Value	Description
31:0	RO	0xb1	The component identification registers identify the PTM as a CoreSight component

**5.4.13 TPIU register summary**

Name	Offset	Size	Reset Value	Description
TPIU_SPSR	0x000	W	0x0000008a	Supported port size
TPIU_CPSR	0x004	W	0x00000001	Current port size
TPIU_STMR	0x100	W	0x11f	Supported trigger modes
TPIU_TCR	0x104	W	0x00	Trigger counter value
TPIU_TMR	0x108	W	0x00	Trigger multiplier
TPIU_STPMR	0x200	W	0x3000f	Supported test pattern/modes
TPIU_CTPMR	0x204	W	0x00000	Current test pattern mode
TPIU_TTPRCCR	0x208	W	0x00	TPIU Test pattern repeat counter
TPIU_FFSR	0x300	W	0x00000006	Formatter and flush status
TPIU_FFCR	0x304	W	0x00000100	Formatter and flush control

TPIU_FSCR	0x308	W	0x00000000	Formatter synchronization counter
TPIU_EXCTLPR_IN	0x400	W	N/A	EXTCTL In Port
TPIU_EXCTLPR_OUT	0x404	W	0x00	EXTCTL Out Port
TPIU_ITTRFLIN_ACK	0xee4	W	N/A	Integration Register
TPIU_ITTRFLIN	0xee8	W	N/A	Integration Register
TPIU_ITATBDA_TA0	0xeeec	W	N/A	Integration Register
TPIU_ITATBCT_R2	0xef0	W	N/A	Integration Register
TPIU_ITATBCT_R0	0xef8	W	N/A	Integration Register
TPIU_ITAMCTL	0xf00	W	0x00000000	Integration Mode control register
TPIU_CTS	0xfa0	W	0x0000000f	Claim tag set
TPIU_CTC	0xfa4	W	0x00000000	Claim tag clear
TPIU_LA	0xfb0	W	N/A	Lock access
TPIU_LS	0xfb4	W	N/A	Local status
TPIU_AS	0xfb8	W	0x00000000	Authentication status
TPIU_DID	0xfc8	W	0x00000000	Device ID
TPIU_DTID	0xfcfc	W	0x00000011	Device type identifier
TPIU_PID4	0xfd0	W	0x00000004	Peripheral ID4
TPIU_PID0	0xfe0	W	0x00000041	Peripheral ID0
TPIU_PID1	0xfe4	W	0x000000b9	Peripheral ID1
TPIU_PID2	0xfe8	W	0x0000000b	Peripheral ID2
TPIU_PID3	0xfec	W	0x00000000	Peripheral ID3
TPIU_CID0	0xff0	W	0x0000000d	Component ID0
TPIU_CID1	0xff4	W	0x00000090	Component ID1

#### 5.4.14 TPIU detailed register description

##### TPIU\_SPSR

Address: TPIU\_BASE + offset(0x000)

Bits	Attr	Reset Value	Description
31:0	RW	0x0000008a	This register is read/write. Each bit location represents a single port size that is supported on the device, that is, 32-1 in bit locations [31:0]. If the bit is set then that port size is allowed. This is to ensure that tools do not attempt to select a port width that cannot be captured by an attached TPA. The value on TPMAXDATASIZE causes bits within the Supported Port Size register that represent wider widths to be clear, that is, unsupported.

##### TPIU\_CPSR

Address: TPIU\_BASE + offset(0x004)

Bits	Attr	Reset Value	Description
31:0	RW	0x00000001	This register is read/write. The Current Port Size Register has the same format as the Supported Port Sizes register but only one bit

			is set, and all others must be zero. Writing values with more than one bit set or setting a bit that is not indicated as supported is not supported and causes unpredictable behavior. On reset this defaults to the smallest possible port size, 1 bit, and so reads as 0x00000001
--	--	--	--

**TPIU\_STMR**

Address: TPIU\_BASE + offset(0x100)

Bits	Attr	Reset Value	Description
31:18	-	-	Reserved
17	RO	0x0	Trigger Counter running. A trigger has occurred but the counter is not at zero.
16	RO	0x1	Triggered. A trigger has occurred and the counter has reached zero
15:9	-	-	Reserved
8	RO	0x0	8-bit wide counter register implemented.
7:5	-	-	Reserved
4	RO	0x1	Multiply the Trigger Counter by 65536 supported.
3	RO	0x1	Multiply the Trigger Counter by 256 supported.
2	RO	0x1	Multiply the Trigger Counter by 16 supported.
1	RO	0x1	Multiply the Trigger Counter by 4 supported
0	RO	0x1	Multiply the Trigger Counter by 2 supported

**TPIU\_TCR**

Address: TPIU\_BASE + offset(0x104)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x0	8-bit counter value for the number of words to be output from the formatter before a trigger is inserted. Reset value is 0x00.

**TPIU\_TMR**

Address: TPIU\_BASE + offset(0x108)

Bits	Attr	Reset Value	Description
31:5	-	-	Reserved
4	RO	0x0	Multiply the Trigger Counter by 65536 supported.
3	RO	0x0	Multiply the Trigger Counter by 256 supported.
2	RO	0x0	Multiply the Trigger Counter by 16 supported.
1	RO	0x0	Multiply the Trigger Counter by 4 supported
0	RO	0x0	Multiply the Trigger Counter by 2 supported

**TPIU\_STPMR**

Address: TPIU\_BASE + offset(0x200)

Bits	Attr	Reset Value	Description
31:18	-	-	Reserved
17	RO	0x1	Continuous mode

16	RO	0x1	Timed mode
15:4	-	-	Reserved
3	RO	0x1	FF/00 Pattern
2	RO	0x1	AA/55 Pattern
1	RO	0x1	Walking 0s Pattern
0	RO	0x1	Walking 1s Pattern

**TPIU\_CTPMR**

Address: TPIU\_BASE + offset(0x204)

Bits	Attr	Reset Value	Description
31:18	-	-	Reserved
17:16	RW	0x0	Mode select
15:4	-	-	Reserved
3:0	RW	0x0	Number of cycles

**TPIU\_TTPRCR**

Address: TPIU\_BASE + offset(0x208)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x0	8-bit counter value to indicate the number of TRACECLKIN cycles that a pattern runs for before switching to the next pattern. Default value is 0.

**TPIU\_FFSR**

Address: TPIU\_BASE + offset(0x300)

Bits	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RO	0x1	If this bit is set then TRACECTL is present. If no TRACECTL pin is available, that is, this bit is zero, then the data formatter must be used and only in continuous mode. If either constraint reports zero/LOW then no TRACECTL is present and this inability to use the pin is reflected in this register.
1	RO	0x1	Formatter stopped. The formatter has received a stop request signal and all trace data and post-amble has been output. Any more trace data on the ATB interface is ignored and ATREADYS goes HIGH.
0	RO	0x0	Flush In Progress. This is an indication of the current state of AFVALIDS

**TPIU\_FFCR**

Address: TPIU\_BASE + offset(0x304)

Bits	Attr	Reset Value	Description
31:14	-	-	Reserved
13	RW	0x0	Stop the formatter after a Trigger Event is observed. Reset to disabled, or zero.
12	RW	0x0	Stop the formatter after a flush completes (return of ATREADYS). This forces the FIFO to drain off any part-completed packets. Setting

			this bit enables this function but this is clear on reset, or disabled.
11	-	-	Reserved
10	RW	0x0	Indicates a trigger on Flush completion on AFREADYS being returned.
9	RW	0x0	Indicate a trigger on a Trigger Event
8	RW	0x1	Indicate a trigger on TRIGIN being asserted.
7	-	-	Reserved
6	RW	0x0	Manually generate a flush of the system. Setting this bit causes a flush to be generated. This is cleared when this flush has been serviced. This bit is clear on reset.
5	RW	0x0	Generate flush using Trigger event. Set this bit to cause a flush of data in the system when a Trigger Event occurs. Reset value is this bit clear.
4	RW	0x0	Generate flush using the FLUSHIN interface. Set this bit to enable use of the FLUSHIN connection. This is clear on reset.
3:2	-	-	Reserved
1	RW	0x0	Continuous Formatting, no TRACECTL. Embed in trigger packets and indicate null cycles using Sync packets. Reset value is this bit clear. Can only be changed when FtStopped is HIGH.
0	RW	0x0	Enable Formatting. Do not embed Triggers into the formatted stream. Trace disable cycles and triggers are indicated by TRACECTL, where fitted. Reset value is this bit clear. Can only be changed when FtStopped is HIGH.

**TPIU\_FSCR**

Address: TPIU\_BASE + offset(0x308)

Bits	Attr	Reset Value	Description
31:12	-	-	Reserved
11:0	RW	0x0	12-bit counter value to indicate the number of complete frames between full synchronization packets. Default value is 64 (0x40).

**TPIU\_ITTRFLINACK**

Address: TPIU\_BASE + offset(0xee4)

Bits	Attr	Reset Value	Description
31:2	-	-	Reserved
1	WO	-	Set the value of FLUSHINACK
0	WO	-	Set the value of TRIGINACK

**TPIU\_ITTRFLIN**

Address: TPIU\_BASE + offset(0xee8)

Bits	Attr	Reset Value	Description
31:2	-	-	Reserved
1	RO	0x0	Read the value of FLUSHIN

0	RO	0x0	Read the value of TRIGIN
---	----	-----	--------------------------

**TPIU\_ITATBDATA0**

Address: TPIU\_BASE + offset(0xeec)

Bits	Attr	Reset Value	Description
31:5	-	-	Reserved
4	RO	0x0	Read the value of ATDATAS[31]
3	RO	0x0	Read the value of ATDATAS[23]
2	RO	0x0	Read the value of ATDATAS[15]
1	RO	0x0	Read the value of ATDATAS[7]
0	RO	0x0	Read the value of ATDATAS[0]

**TPIU\_ITATBCTR2**

Address: TPIU\_BASE + offset(0xef0)

Bits	Attr	Reset Value	Description
31:2	-	-	Reserved
1	WO	0x0	Set the value of AFVALIDS
0	WO	0x0	Set the value of ATREADYS

**TPIU\_ITATBCTR1**

Address: TPIU\_BASE + offset(0xef4)

Bits	Attr	Reset Value	Description
31:7	-	-	Reserved
6:0	RO	0x0	Read the value of ATIDS

**TPIU\_ITATBCTR0**

Address: TPIU\_BASE + offset(0xef8)

Bits	Attr	Reset Value	Description
31:10	-	-	Reserved
9:8	RO	0x0	Read the value of ATBYTESS
7:2	-	-	Reserved
1	RO	0x0	Read the value of AFREADYS
0	RO	0x0	Read the value of ATVALIDS

**TPIU\_CTS-TPIU\_CTC**

Address: TPIU\_BASE + offset(0xfa0-0xfa4)

Bits	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RW	0x0	The TPIU implements a four-bit claim tag. The use of bits [3:0] is software defined

**TPIU\_LA-TPIU\_LS**

Address: TPIU\_BASE + offset(0xfb0-0xfb4)

Bits	Attr	Reset Value	Description
31:3	-	0xb1	Reserved
2:0	RW	0x3	The TPIU implements two memory maps controlled through PADDRDBG31. When PADDRDBG31 is HIGH, the Lock Status Register reads as 0x0 indicating that no lock exists. When PADDRDBG31 is LOW, the Lock

			Status Register reads as 0x3 from reset. This indicates a 32-bit lock access mechanism is present and is locked.
--	--	--	--

**TPIU\_AS**

Address: TPIU\_BASE + offset(0xfb8)

Bits	Attr	Reset Value	Description
31:0	RO	0x0	Reports the required security level. The TPIU has a default value of 0x00 to indicate that this functionality is not implemented.

**TPIU\_DID**

Address: TPIU\_BASE + offset(0xfc8)

Bits	Attr	Reset Value	Description
31:12	-	-	Reserved
11	RO	0x0	Indicates Serial Wire Output (UART/NRZ) is not supported.
10	RO	0x0	Indicates Serial Wire Output (Manchester) is not supported.
9	RO	0x0	Indicates trace clock + data is supported.
8:6	RO	0x0	FIFO size in powers of 2. A value of 2 gives a FIFO size of 4 entries, 16 bytes.
5	RO	0x0	Indicates the relationship between ATCLK and TRACECLKIN. 0x1 indicates asynchronous.
4:0	RO	0x0	Hidden Level of Input multiplexing. When nonzero this value indicates the type/number of ATB multiplexing present on the input to the ATB. Currently only 0x00 is supported, that is, no multiplexing present. This value is used to assist topology detection of the ATB structure

**TPIU\_DTIID**

Address: TPIU\_BASE + offset(0xfc0)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x11	0x11 indicates this device is a trace sink (0x1) and specifically a TPIU (0x1).

**TPIU\_PID4**

Address: TPIU\_BASE + offset(0xfd0)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
3:0	RO	0x3	The TPIU is identified as an ARM component with a JEP106 identity at 0x3B and a JEP106 continuation code at 0x4 (fifth bank).

**TPIU\_PID0**

Address: TPIU\_BASE + offset(0xfe0)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:4	RO	0x1	Middle BCD value of Device number.

3:0	RO	0x2	Lower BCD value of Device number.
-----	----	-----	-----------------------------------

**TPIU\_PID1**

Address: TPIU\_BASE + offset(0xfe4)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:4	RO	0xb	The TPIU is identified as an ARM component with a JEP106 identity at 0x3B and a JEP106 continuation code at 0x4 (fifth bank).
3:0	RO	0x9	Upper BCD value of Device number.

**TPIU\_PID2**

Address: TPIU\_BASE + offset(0xfe8)

Bits	Attr	Reset Value	Description
31:3	-	-	Reserved
2:0	RO	0x4	The TPIU is identified as an ARM component with a JEP106 identity at 0x3B and a JEP106 continuation code at 0x4 (fifth bank).
3:0	-	-	Reserved

**TPIU\_PID3**

Address: TPIU\_BASE + offset(0xfec)

Bits	Attr	Reset Value	Description
31:8	-	-	Reserved
7:4	RO	0x00	The peripheral identification registers provide standard information required for all CoreSight components.
3:0	-	-	Reserved

**TPIU\_CID1**

Address: TPIU\_BASE + offset(0xff4)

Bits	Attr	Reset Value	Description
31:8	RO	0x0	Reserved
7:4	RO	0x9	The TPIU complies to the CoreSight class of components and this value is set to 0x9.
3:0	-	0x0	Reserved

Notes: Attr: **RW**- Read/writable, **RO**- read only, **WO**- write only, **RWTC**-Readable and write "1" to clear the asserted bit from "1" to "0".

## 5.5 Interface description

### 5.5.1 DAP SWJ-DP interface

The following figure is the DAP SWJ-DP interface, the SWJ-DP is a combined JTAG-DP and SW-DP that enable you connect either a Serial Write Debug(SWJ) to JTAG probe to a target.

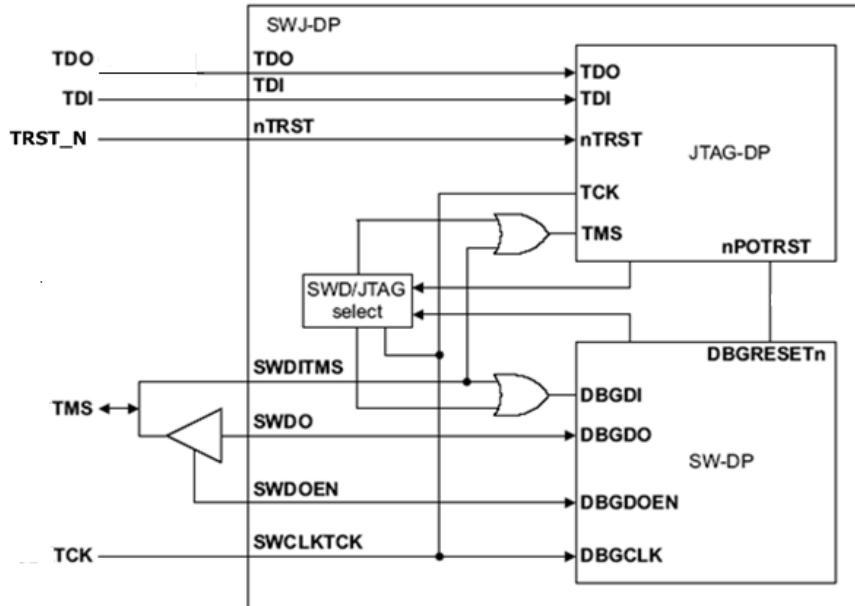


Fig. 5-3 DAP SWJ interface

Table 5-1 SWJ interface

<b>Module Pin</b>	<b>Direction</b>	<b>PAD Name</b>	<b>IOMUX Setting</b>
TRST_N	I	IO_SDMMC0data1_JTA Gtrstn_SDCARDgpio6c1	GRF_GPIO6C_IOMUX[9:0]= 0x2aa
TCK	I	IO_SDMMC0data3_JTA Gtck_SDCARDgpio6c3	GRF_GPIO6C_IOMUX[9:0]= 0x2aa
TDI	I	IO_SDMMC0data2_JTA Gtdi_SDCARDgpio6c2	GRF_GPIO6C_IOMUX[9:0]= 0x2aa
TMS	IO	IO_SDMMC0data0_JTA Gtms_SDCARDgpio6c0	GRF_GPIO6C_IOMUX[9:0]= 0x2aa
TDO	O	IO_SDMMC0clkout_JTA Gtdo_SDCARDgpio6c4	GRF_GPIO6C_IOMUX[9:0]= 0x2aa

### 5.5.2 TPIU trace port interface

Table 5-2 TPIU interface

<b>Module Pin</b>	<b>Direction</b>	<b>PAD Name</b>	<b>IOMUX Setting</b>
trace_data[0]	O	IO_LVDS_DATAP0	GRF_SOC_CON7[13]=0x1
trace_data[1]	O	IO_LVDS_DATAN0	GRF_SOC_CON7[13]=0x1
trace_data[2]	O	IO_LVDS_DATAP1	GRF_SOC_CON7[13]=0x1
trace_data[3]	O	IO_LVDS_DATAN1	GRF_SOC_CON7[13]=0x1
trace_data[4]	O	IO_LVDS_DATAP2	GRF_SOC_CON7[13]=0x1
trace_data[5]	O	IO_LVDS_DATAN2	GRF_SOC_CON7[13]=0x1
trace_data[6]	O	IO_LVDS_DATAP3	GRF_SOC_CON7[13]=0x1
trace_data[7]	O	IO_LVDS_DATAN3	GRF_SOC_CON7[13]=0x1
trace_data[8]	O	IO_LVDS_DATAP4	GRF_SOC_CON7[13]=0x1
trace_data[9]	O	IO_LVDS_DATAN4	GRF_SOC_CON7[13]=0x1
trace_data[10]	O	IO_LVDS_CLKP0	GRF_SOC_CON7[13]=0x1
trace_data[11]	O	IO_LVDS_CLKN0	GRF_SOC_CON7[13]=0x1

trace_data[12]	O	IO_LVDS_DATAP5	GRF_SOC_CON7[13]=0x1
trace_data[13]	O	IO_LVDS_DATAN5	GRF_SOC_CON7[13]=0x1
trace_data[14]	O	IO_LVDS_DATAP6	GRF_SOC_CON7[13]=0x1
trace_data[15]	O	IO_LVDS_DATAN6	GRF_SOC_CON7[13]=0x1
trace_clk	O	IO_LVDS_DATAP7	GRF_SOC_CON7[13]=0x1
trace_ctl	O	IO_LVDS_DATAN7	GRF_SOC_CON7[13]=0x1

## Chapter 6 System Security

### 6.1 Overview

The RK3288 uses the TrustZone access control scheme to support the system security application requirement.

### 6.2 Block Diagram

The following figure is the system security architecture. All the devices which support security access are demonstrated in this figure.

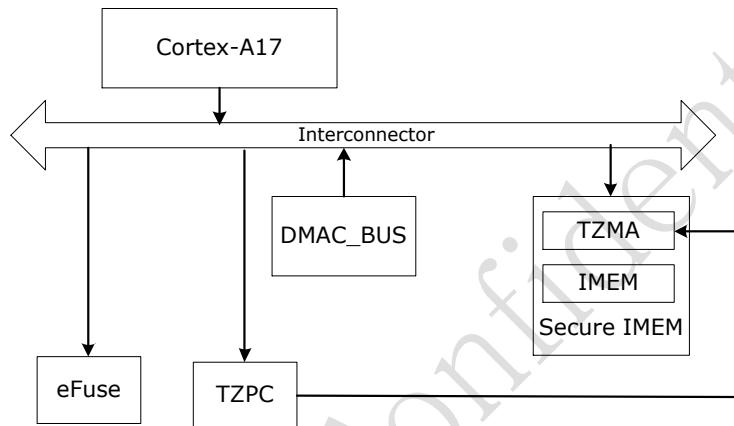


Fig. 6-1 RK3288 security architecture

### 6.3 Function Description

#### 6.3.1 Cortex-A17 Security Extension architecture

The processor implements the TrustZone Security Extensions architecture to facilitate the development of secure applications.

Security Extensions are based on these fundamental principles:

- The extensions define a class of core operation that you can switch between secure and non-secure state. Most code runs in non-secure state. Only trusted code runs in secure state
- The extensions define some memory as secure memory. When the core is in secure state, it can access secure memory
- Entry into secure state is strictly controlled
- Exit from secure state can only occur at programmed points
- Debug is strictly controlled
- The processor enters secure state on reset

#### 6.3.2 TZPC

The TZPC (Trust Protection Controller) is the APB slave, which configured by software and control secure memory space of Embedded SRAM and system security device setting. The TZPC support the following feature

- control the secure memory size of secure Embedded SRAM by configuring TZPCR0SIZE
- control DMAC\_BUS Secure-Configuration bit by configuring TZPCDECProt1 and TZPCDECProt2

The following is the TZPC block diagram.

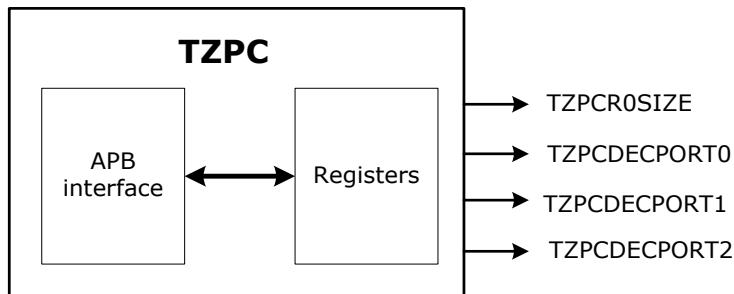


Fig. 6-2 TZPC block diagram

The TZPC can only be accessed on secure world and the software can configure the following items:

- Configure TZPCR0SIZE to set TZMA secure access space, which can be 0KB, 4KB, 8KB, 12KB... up to 96KB by 4KB step
- Configure TZPCDECPORTx to set DMAC\_BUS secure input port

*Note: Please refer to the registers of TZPC\_DECProt0Set, TZPC\_DECProt1Set and TZPC\_DECProt2 Set for detailed information about TZPCDECPORTx.*

### 6.3.3 TZMA

The TZMA (TrustZone Memory Adapter) is a bridge between AXI bus and Embedded SRAM, which support the flexible secure access by controlling R0SIZE port.

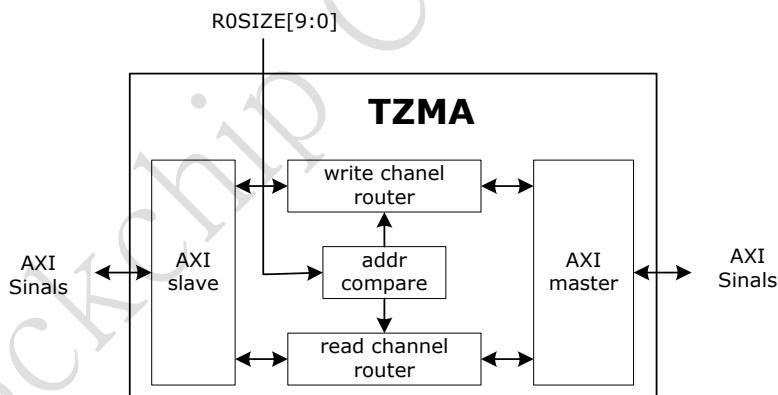


Fig. 6-3 TZMA block diagram

TZMA can support 0KB, 4KB, 8KB, 12KB... up to 96KB by 4KB step(the whole Embedded SRAM space) secure access by setting ROSIZE[9:0].

### 6.3.4 DMAC\_BUS secure access

The DMAC\_BUS is an AMBA compliant peripheral, which provides an AXI interface to perform the DMA transfer and two APB interfaces that control its operation. The DMAC\_BUS implements TrustZone secure technology with one APB interface operating in the secure state and the other operating in the Non-secure state. For the detailed description for DMAC\_BUS, please refer to Chapter 10.

The following diagram shows the interface of DMAC\_BUS.

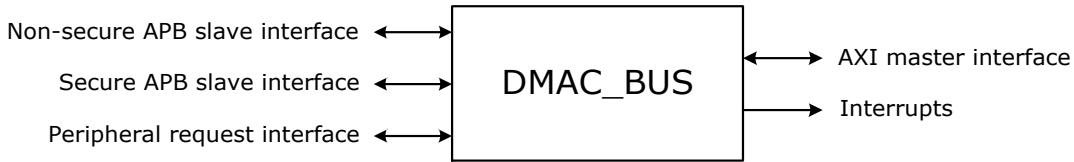


Fig. 6-4 DMAC\_BUS interface

The DMAC\_BUS support the secure feature in the following section:

- DMA manager thread
- DMA channel thread
- Event and interrupts
- Peripheral request interface

1. The security of DMA manager thread is controlled by input port boot\_manager\_ns.

0=assign DMA manager to the Secure state

1=assign DMA manager to the Non-secure state

2. The security of DMA channel thread is controlled by instruction DMAGO ns bit. If ns is present, DMA channel operation is in the non-secure state. Otherwise, the execution of the instruction depends on the security of the state of DMA manager:

DMA manager is in the secure state, DMA channel operates in the secure state  
DMA manager is in the non-secure state, DMA abort

3. The security state of the event-interrupt source is controlled by the input port boot\_irq\_ns[x:0], if boot\_irq\_ns[x] is LOW, the DMAC\_BUS assign event<x> or irq[x] to the secure state, otherwise the DMAC\_BUS assign event<x> or irq[x] to the non-secure state.

4. The security state of peripheral request interface is controlled by the input port boot\_irq\_ns[x], if boot\_irq\_ns[x] is LOW, the DMAC\_BUS assign peripheral interface x to secure state, otherwise the DMAC\_BUS assign peripheral interface x to non-secure state.

### 6.3.5 Bus components security setting

The following table describes the security support on bus components which have the master interface or slave interface.

Table 6-1 bus components security setting

AXI master	secure	All transactions originating from this master interface are flagged as secure transaction and can access both secure and non-secure component.
	non-secure	All transactions originating from this master interface are flagged as non-secure transactions and cannot access secure component
	per access	The AxPROTx signal determines the security setting of each transaction, and the slave that it can access
AHB-Lite master	secure	All transactions originating from this master interface are flagged as secure transaction and can access both secure and non-secure component.
	non-secure	All transactions originating from this master interface are flagged as non-secure transactions and cannot access secure component
AXI slave	secure	Only secure transactions can access this component
	non-secure	Both secure and non-secure transactions can

		access this components
	boot-secure	You can use software to configure whether it permits secure and non-secure transactions to access component. When boot up, this component only can be accessed by secure transactions.
AHB slave	secure	Only secure transactions can access this components
	non-secure	Both secure and non-secure transactions can access this components
	boot-secure	You can use software to configure whether it permit secure and non-secure transactions to access this components. When boot up, this component only can be accessed by secure transactions.
APB slave	secure	Only secure transactions can access this components
	non-secure	Both secure and non-secure transactions can access this components
	boot-secure	You can use software to configure whether it permit secure and non-secure transactions to access this components. When boot up, this component only can be accessed by secure transactions.

Notes: When a non-secure master tries to access a secure slave, an error response will be returned. In RK3288, Cortex-A17 non-secure access a secure slave will cause a data-abort, and dmac\_bus non-secure access a secure slave will cause an interrupt for access error.

### 6.3.6 RK3288 secure device setting

Table 6-2 RK3288 secure device setting

Cortex-A17	AXI master	Per access
TZPC	APB slave	Secure
eFuse	APB slave	Secure
Embedded SRAM	AXI slave	Controlled by TZMA, the secure space can be set to 0, 4KB, 8KB, 12KB ...up to 96KB
DMAC_BUS	AXI master	Per access
	Secure APB slave	Secure
	Non-secure APB slave	Non-secure

### 6.3.7 RK3288 device secure input port setting

The following table lists all the secure input ports for the secure device. These secure input ports could be set by configuring TZPCDECPORTx registers. Please refer to 5.4.2 for detailed information.

Table 6-3 RK3288 device secure input port setting

Input Port	Module	Function description
boot_manager_ns	DMAC_BUS	When the DMAC exits from reset, this signal controls the security state of the DMAmanager thread: 1'b0: assigns DMA manager to the secure state 1'b1: assigns DMA manager to the non-secure state
boot_perih_ns[2:0]	DMAC_BUS	Controls the security state of a peripheral

		request interface, when the DMAC exits from reset: boot_periph_ns[x] is LOW The DMAC assigns peripheral request interface x to the secure state boot_periph_ns[x] is HIGH The DMAC assigns peripheral request interface x to the non-secure state.
boot_irq_ns[7:0]	DMAC_BUS	Controls the security state of an event-interrupt resource, when the DMAC exits from reset: boot_irq_ns[x] is LOW The DMAC assigns event<x> or irq[x] to the secure state. boot_irq_ns[x] is HIGH The DMAC assigns event<x> or irq[x] to the non-secure state.

## 6.4 Register Description

The base address of secure components TZPC is 0xffb00000.

### 6.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
TZPC_ROSIZE	0x000	W	0x00000200	Secure RAM region size register
TZPC_DECPROT1Stat	0x80c	W	0x000000ff	Secure-GRF register status
TZPC_DECPROT1Set	0x810	W	N/A	Secure-GRF register set
TZPC_DECPROT1Clr	0x814	W	N/A	Secure-GRF register clear
TZPC_DECPROT2Stat	0x818	W	0x00000006	Secure-GRF register status
TZPC_DECPROT2Set	0x81c	W	N/A	Secure-GRF register set
TZPC_DECPROT2Clr	0x820	W	N/A	Secure-GRF register clear

### 6.4.2 Detail Register Description

#### TZPC\_ROSIZE

Address :TZPC\_BASE + offset(0x00)

Secure RAM Region Size Register

Bits	Attr	Reset Value	Description
31:10	R	0x0	Reserved
9:0	RW	0x200	Secure RAM region size in 4KB steps 9'b00: no secure region 9'b01: 4KB secure region 9'b10: 8KB secure region 9'b11: 12KB secure region ... 0x200 or above sets the entire Embedded SRAM space to secure

#### TZPC\_DECPROT1Stat

Address :TZPC\_BASE + offset(0x80c)

Status bit for secure device input port control

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	R	0x0	Reserved
7:2	R	0xff	Status bit for DMAC_BUS boot_perih_ns input port control
1:0	R	0x0	Reserved

**TZPC\_DECPROT1Set**

Address :TZPC\_BASE + offset(0x810)

Set bit for secure device input port control

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	W	N/A	Reserved
7:2	W	N/A	Set bit for DMAC_BUS boot_perih_ns input port control
1:0	R	0x0	Reserved

**TZPC\_DECPROT1Clr**

Address :TZPC\_BASE + offset(0x814)

Set bit for secure device input port control

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	W	N/A	Reserved
7:2	W	N/A	Clear bit for DMAC_BUS boot_perih_ns input port control
1:0	R	0x0	Reserved

**TZPC\_DECPROT2Stat**

Address :TZPC\_BASE + offset(0x818)

Status bit for secure device input port control

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	R	0x0	Reserved
7:1	R	0x06	Status bit for DMAC_BUS boot_irq_ns input port control
0	R	0x0	Status bit for DMAC_BUS boot_manage_ns input port control

**TZPC\_DECPROT2Set**

Address :TZPC\_BASE + offset(0x81c)

Set bit for secure device input port control

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	W	N/A	Reserved
7:1	W	N/A	Set bit for DMAC_BUS boot_irq_ns input port control
0	W	N/A	Set bit for DMAC_BUS boot_manage_ns input port control

**TZPC\_DECPROT2Clr**

Address :TZPC\_BASE + offset(0x820)

Set bit for secure device input port control

<b>Bits</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	W	undef	Reserved
7:1	W	undef	Clear bit for DMAC_BUS boot_irq_ns input port control
0	W	undef	Clear bit for DMAC_BUS boot_manage_ns input port control

## 6.5 Application Notes

### Secure software conception

The basis of the security extensions model is that the computing environment splits into two isolated states, the secure state and the non-secure state, with no leakage of secure data to the non-secure state. Software secure monitor code, running in the monitor mode, links the two states and acts as a gatekeeper to manage program flow. The system can have both secure and non-secure peripherals that are suitable to secure and non-secure device driver control. Following figure shows the relationship between the secure and non-secure states. The operating system (OS) splits into the secure OS, that includes the secure kernel, and the non-secure OS, that includes the non-secure kernel.

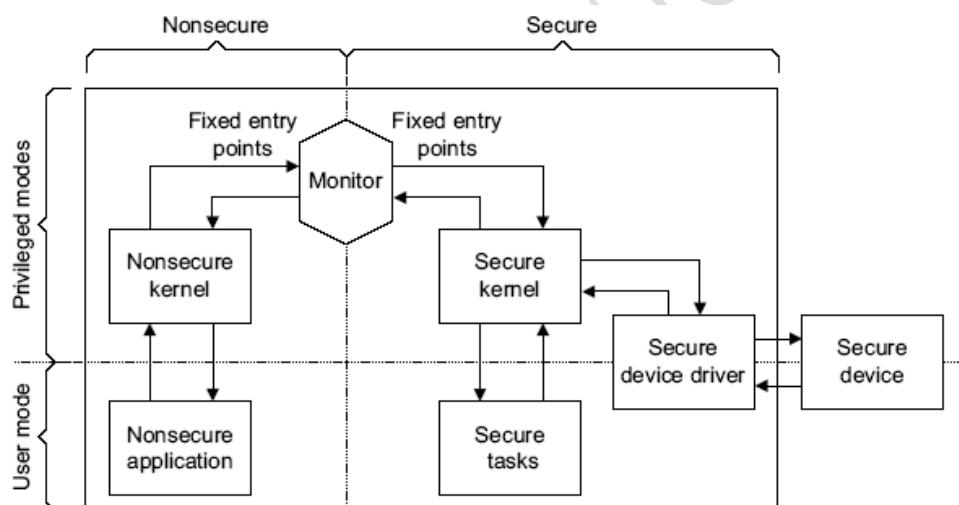


Fig. 6-5 Software Diagram of Secure and Non-secure

In normal non-secure operation, the OS runs tasks in the usual way. When a user process requires secure execution it makes a request to the secure kernel, that operates in privileged mode. This then calls the secure monitor to transfer execution to the secure state.

This approach to secure systems means that the platform OS that works in the non-secure state, has only a few fixed entry points into the secure state through the secure monitor. The trusted code base for the secure state, that includes the secure kernel and secure device drivers, is small and therefore much easier to maintain and verify.

### Secure/Non-secure memory space for Embedded SRAM

The following figure gives an example of embedded SRAM secure/non-secure memory space setting. The software configures 4KB secure space by setting TZPC\_R0SIZE=0x1. The bottom 4KB space will act as secure space and the other 92K space will be non-secure space.

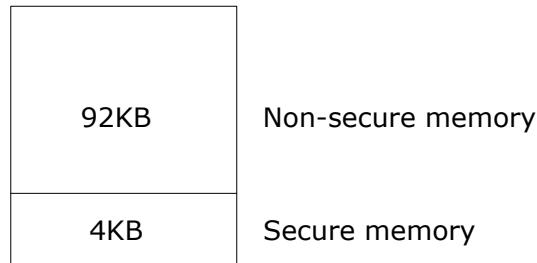


Fig. 6-6 Embedded SRAM secure memory space setting

Rockchip Confidential

## Chapter 7 General Register Files (GRF)

### 7.1 Overview

The general register file will be used to do static set by software, which is composed of many registers for system control. The GRF is divided into two sections, one is GRF for non-secure system, the other is SGRF for secure system.

### 7.2 Function Description

The function of general register file is:

- IOMUX control
- Control the state of GPIO in power-down mode
- GPIO PAD pull down and pull up control
- Used for common system control
- Used to record the system state

### 7.3 GRF Register Description

#### 7.3.1 Register Summary

Name	Offset	Size	Reset Value	Description
GRF_GPIO1D_IOMUX	0x000c	W	0x00000000	GPIO1D iomux control
GRF_GPIO2A_IOMUX	0x0010	W	0x00000000	GPIO2A iomux control
GRF_GPIO2B_IOMUX	0x0014	W	0x00000000	GPIO2B iomux control
GRF_GPIO2C_IOMUX	0x0018	W	0x00000000	GPIO2C iomux control
GRF_GPIO3A_IOMUX	0x0020	W	0x00000000	GPIO3A iomux control
GRF_GPIO3B_IOMUX	0x0024	W	0x00000000	GPIO3B iomux control
GRF_GPIO3C_IOMUX	0x0028	W	0x00000000	GPIO3C iomux control
GRF_GPIO3DL_IOMUX	0x002c	W	0x00000000	GPIO3D iomux control
GRF_GPIO3DH_IOMUX	0x0030	W	0x00000000	GPIO3D iomux control
GRF_GPIO4AL_IOMUX	0x0034	W	0x00000000	GPIO4A iomux control
GRF_GPIO4AH_IOMUX	0x0038	W	0x00000000	GPIO4A iomux control
GRF_GPIO4BL_IOMUX	0x003c	W	0x00000000	GPIO4B iomux control
GRF_GPIO4C_IOMUX	0x0044	W	0x00000000	GPIO4C iomux control
GRF_GPIO4D_IOMUX	0x0048	W	0x00000000	GPIO4D iomux control
GRF_GPIO5B_IOMUX	0x0050	W	0x00000000	GPIO5B iomux control
GRF_GPIO5C_IOMUX	0x0054	W	0x00000000	GPIO5C iomux control
GRF_GPIO6A_IOMUX	0x005c	W	0x00000000	GPIO6A iomux control
GRF_GPIO6B_IOMUX	0x0060	W	0x00000000	GPIO6B iomux control
GRF_GPIO6C_IOMUX	0x0064	W	0x00001555	GPIO6C iomux control
GRF_GPIO7A_IOMUX	0x006c	W	0x00000000	GPIO7A iomux control
GRF_GPIO7B_IOMUX	0x0070	W	0x00000000	GPIO7B iomux control
GRF_GPIO7CL_IOMUX	0x0074	W	0x00000000	GPIO7CL iomux control
GRF_GPIO7CH_IOMUX	0x0078	W	0x00000000	GPIO7CH iomux control

Name	Offset	Size	Reset Value	Description
GRF_GPIO8A_IOMUX	0x0080	W	0x00000000	GPIO8A iomux control
GRF_GPIO8B_IOMUX	0x0084	W	0x00000000	GPIO8B iomux control
GRF_GPIO1H_SR	0x0104	W	0x00000f00	GPIO1C/D SR control
GRF_GPIO2L_SR	0x0108	W	0x00000000	GPIO2A/B SR control
GRF_GPIO2H_SR	0x010c	W	0x00000000	GPIO2C/D SR control
GRF_GPIO3L_SR	0x0110	W	0x000020ff	GPIO3A/B SR control
GRF_GPIO3H_SR	0x0114	W	0x0000ff04	GPIO3C/D SR control
GRF_GPIO4L_SR	0x0118	W	0x00000120	GPIO4A/B SR control
GRF_GPIO4H_SR	0x011c	W	0x00000000	GPIO4C/D SR control
GRF_GPIO5L_SR	0x0120	W	0x00000000	GPIO5A/B SR control
GRF_GPIO5H_SR	0x0124	W	0x00000000	GPIO5C/D SR control
GRF_GPIO6L_SR	0x0128	W	0x00000100	GPIO6A/B SR control
GRF_GPIO6H_SR	0x012c	W	0x00000010	GPIO6C/D SR control
GRF_GPIO7L_SR	0x0130	W	0x00000000	GPIO7A/B SR control
GRF_GPIO7H_SR	0x0134	W	0x00000000	GPIO7C/D SR control
GRF_GPIO8L_SR	0x0138	W	0x00000000	GPIO8A/B SR control
GRF_GPIO1D_P	0x014c	W	0x0000aaaa	GPIO1D PU/PD control
GRF_GPIO2A_P	0x0150	W	0x0000aaaa	GPIO2A PU/PD control
GRF_GPIO2B_P	0x0154	W	0x0000aaaa	GPIO2B PU/PD control
GRF_GPIO2C_P	0x0158	W	0x0000aaa5	GPIO2C PU/PD control
GRF_GPIO3A_P	0x0160	W	0x00005555	GPIO3A PU/PD control
GRF_GPIO3B_P	0x0164	W	0x00005699	GPIO3B PU/PD control
GRF_GPIO3C_P	0x0168	W	0x0000aaa5	GPIO3C PU/PD control
GRF_GPIO3D_P	0x016c	W	0x00005555	GPIO3D PU/PD control
GRF_GPIO4A_P	0x0170	W	0x00005555	GPIO4A PU/PD control
GRF_GPIO4B_P	0x0174	W	0x0000aaa5	GPIO4B PU/PD control
GRF_GPIO4C_P	0x0178	W	0x00005559	GPIO4C PU/PD control
GRF_GPIO4D_P	0x017c	W	0x00005a99	GPIO4D PU/PD control
GRF_GPIO5B_P	0x0184	W	0x00006559	GPIO5B PU/PD control
GRF_GPIO5C_P	0x0188	W	0x0000aaa9	GPIO5C PU/PD control
GRF_GPIO6A_P	0x0190	W	0x0000aaaa	GPIO6A PU/PD control
GRF_GPIO6B_P	0x0194	W	0x0000aa96	GPIO6B PU/PD control
GRF_GPIO6C_P	0x0198	W	0x00005655	GPIO6C PU/PD control
GRF_GPIO7A_P	0x01a0	W	0x000059aa	GPIO7A PU/PD control
GRF_GPIO7B_P	0x01a4	W	0x0000a696	GPIO7B PU/PD control
GRF_GPIO7C_P	0x01a8	W	0x00005955	GPIO7C PU/PD control
GRF_GPIO8A_P	0x01b0	W	0x00006555	GPIO8A PU/PD control
GRF_GPIO8B_P	0x01b4	W	0x0000aaaa	GPIO8B PU/PD control
GRF_GPIO1D_E	0x01cc	W	0x000055aa	GPIO1D drive strength control
GRF_GPIO2A_E	0x01d0	W	0x0000aaaa	GPIO2A drive strength control
GRF_GPIO2B_E	0x01d4	W	0x0000aaaa	GPIO2B drive strength control
GRF_GPIO2C_E	0x01d8	W	0x00005555	GPIO2C drive strength control
GRF_GPIO3A_E	0x01e0	W	0x0000aaaa	GPIO3A drive strength control

Name	Offset	Size	Reset Value	Description
GRF_GPIO3B_E	0x01e4	W	0x00005955	GPIO3B drive strength control
GRF_GPIO3C_E	0x01e8	W	0x00005565	GPIO3C drive strength control
GRF_GPIO3D_E	0x01ec	W	0x0000aaaa	GPIO3D drive strength control
GRF_GPIO4A_E	0x01f0	W	0x00005955	GPIO4A drive strength control
GRF_GPIO4B_E	0x01f4	W	0x00005556	GPIO4B drive strength control
GRF_GPIO4C_E	0x01f8	W	0x00005555	GPIO4C drive strength control
GRF_GPIO4D_E	0x01fc	W	0x00005555	GPIO4D drive strength control
GRF_GPIO5B_E	0x0204	W	0x00005555	GPIO5B drive strength control
GRF_GPIO5C_E	0x0208	W	0x00005555	GPIO5C drive strength control
GRF_GPIO6A_E	0x0210	W	0x00005555	GPIO6A drive strength control
GRF_GPIO6B_E	0x0214	W	0x00005555	GPIO6B drive strength control
GRF_GPIO6C_E	0x0218	W	0x00005555	GPIO6C drive strength control
GRF_GPIO7A_E	0x0220	W	0x00005555	GPIO7A drive strength control
GRF_GPIO7B_E	0x0224	W	0x00005555	GPIO7B drive strength control
GRF_GPIO7C_E	0x0228	W	0x00005555	GPIO7C drive strength control
GRF_GPIO8A_E	0x0230	W	0x00005555	GPIO8A drive strength control
GRF_GPIO8B_E	0x0234	W	0x00005555	GPIO8B drive strength control
GRF_GPIO_SMT	0x0240	W	0x00000fff	GPIO smitter control register
GRF_SOC_CON0	0x0244	W	0x00001c18	SoC control register 0
GRF_SOC_CON1	0x0248	W	0x00004040	SoC control register 1
GRF_SOC_CON2	0x024c	W	0x00000002	SoC control register 2
GRF_SOC_CON3	0x0250	W	0x00000810	SoC control register 3
GRF_SOC_CON4	0x0254	W	0x00000607	SoC control register 4
GRF_SOC_CON5	0x0258	W	0x00008c87	SoC control register 5
GRF_SOC_CON6	0x025c	W	0x00008000	SoC control register 6
GRF_SOC_CON7	0x0260	W	0x00000000	SoC control register 7
GRF_SOC_CON8	0x0264	W	0x0000000e	SoC control register 8
GRF_SOC_CON9	0x0268	W	0x0000000e	SoC control register 9
GRF_SOC_CON10	0x026c	W	0x0000000f	SoC control register 10
GRF_SOC_CON11	0x0270	W	0x00000000	SoC control register 11
GRF_SOC_CON12	0x0274	W	0x00000013	SoC control register 12
GRF_SOC_CON13	0x0278	W	0x00000000	SoC control register 13
GRF_SOC_CON14	0x027c	W	0x00000000	SoC control register 14
GRF_SOC_STATUS0	0x0280	W	0x00000000	SoC status register 0
GRF_SOC_STATUS1	0x0284	W	0x00000000	SoC status register 1
GRF_SOC_STATUS2	0x0288	W	0x00000000	SoC status register 2
GRF_SOC_STATUS3	0x028c	W	0x00000000	SoC status register 3
GRF_SOC_STATUS4	0x0290	W	0x00000000	SoC status register 4
GRF_SOC_STATUS5	0x0294	W	0x00000000	SoC status register 5
GRF_SOC_STATUS6	0x0298	W	0x00000000	SoC status register 6
GRF_SOC_STATUS7	0x029c	W	0x00000000	SoC status register 7
GRF_SOC_STATUS8	0x02a0	W	0x00000000	SoC status register 8
GRF_SOC_STATUS9	0x02a4	W	0x00000000	SoC status register 9

Name	Offset	Size	Reset Value	Description
GRF_SOC_STATUS10	0x02a8	W	0x00000000	SoC status register 10
GRF_SOC_STATUS11	0x02ac	W	0x00000000	SoC status register 11
GRF_SOC_STATUS12	0x02b0	W	0x00000000	SoC status register 12
GRF_SOC_STATUS13	0x02b4	W	0x00000000	SoC status register 13
GRF_SOC_STATUS14	0x02b8	W	0x00000000	SoC status register 14
GRF_SOC_STATUS15	0x02bc	W	0x00000000	SoC status register 15
GRF_SOC_STATUS16	0x02c0	W	0x00000000	SoC status register 16
GRF_SOC_STATUS17	0x02c4	W	0x00000000	SoC status register 17
GRF_SOC_STATUS18	0x02c8	W	0x00000000	SoC status register 18
GRF_SOC_STATUS19	0x02cc	W	0x00000000	SoC status register 19
GRF_SOC_STATUS20	0x02d0	W	0x00000000	SoC status register 20
GRF_SOC_STATUS21	0x02d4	W	0x00000000	SoC status register 21
GRF_PERIDMAC_CON0	0x02e0	W	0x000000fa	PERI DMAC control register 0
GRF_PERIDMAC_CON1	0x02e4	W	0x00000000	PERI DMAC control register 1
GRF_PERIDMAC_CON2	0x02e8	W	0x0000ffff	PERI DMAC control register 2
GRF_PERIDMAC_CON3	0x02ec	W	0x0000ffff	PERI DMAC control register 3
GRF_DDRC0_CON0	0x02f0	W	0x00000000	DDRC0 control register 0
GRF_DDRC1_CON0	0x02f4	W	0x00000000	DDRC1 control register 0
GRF_CPU_CON0	0x02f8	W	0x00008220	CPU control register 0
GRF_CPU_CON1	0x02fc	W	0x00000ff0	CPU control register 1
GRF_CPU_CON2	0x0300	W	0x00000fff	CPU control register 2
GRF_CPU_CON3	0x0304	W	0x00000000	CPU control register 3
GRF_CPU_CON4	0x0308	W	0x00002400	CPU control register 4
GRF_CPU_STATUS0	0x0318	W	0x00000000	CPU status register 0
GRF_UOC0_CON0	0x0320	W	0x00000089	UOC0 control register 0
GRF_UOC0_CON1	0x0324	W	0x00007333	UOC0 control register 1
GRF_UOC0_CON2	0x0328	W	0x00000d08	UOC0 control register 2
GRF_UOC0_CON3	0x032c	W	0x00000001	UOC0 control register 3
GRF_UOC0_CON4	0x0330	W	0x00000003	UOC0 control register 4
GRF_UOC1_CON0	0x0334	W	0x00000b89	UOC1 control register 0
GRF_UOC1_CON1	0x0338	W	0x00007333	UOC1 control register 1
GRF_UOC1_CON2	0x033c	W	0x00000d08	UOC1 control register 2
GRF_UOC1_CON3	0x0340	W	0x00001c41	UOC1 control register 3
GRF_UOC1_CON4	0x0344	W	0x0000c820	UOC1 control register 4
GRF_UOC2_CON0	0x0348	W	0x00000089	UOC2 control register 0
GRF_UOC2_CON1	0x034c	W	0x00007333	UOC2 control register 1
GRF_UOC2_CON2	0x0350	W	0x00000d08	UOC2 control register 2
GRF_UOC2_CON3	0x0354	W	0x00001c01	UOC2 control register 3
GRF_UOC3_CON0	0x0358	W	0x000030eb	UOC3 control register 0
GRF_UOC3_CON1	0x035c	W	0x00000003	UOC3 control register 1
GRF_UOC4_CON0	0x0360	W	0x00001080	UOC4 control register 0
GRF_UOC4_CON1	0x0364	W	0x00000820	UOC4 control register 1
GRF_PVTM_CON0	0x0368	W	0x00000000	PVT monitor control register 0

Name	Offset	Size	Reset Value	Description
GRF_PVTM_CON1	0x036c	W	0x016e3600	PVT monitor control register 1
GRF_PVTM_CON2	0x0370	W	0x016e3600	PVT monitor control register 2
GRF_PVTM_STATUS0	0x0374	W	0x00000000	PVT monitor status register 0
GRF_PVTM_STATUS1	0x0378	W	0x00000000	PVT monitor status register 1
GRF_PVTM_STATUS2	0x037c	W	0x00000000	PVT monitor status register 2
GRF_IO_VSEL	0x0380	W	0x00000004	IO voltage select
GRF_SARADC_TESTBIT	0x0384	W	0x00000000	SARADC Test bit register
GRF_TSADC_TESTBIT_L	0x0388	W	0x00000000	TSADC Test bit low register
GRF_TSADC_TESTBIT_H	0x038c	W	0x00000000	TSADC Test bit high register
GRF_OS_REG0	0x0390	W	0x00000000	OS register 0
GRF_OS_REG1	0x0394	W	0x00000000	OS register 1
GRF_OS_REG2	0x0398	W	0x00000000	OS register 2
GRF_OS_REG3	0x039c	W	0x00000000	OS register 3
GRF_SOC_CON15	0x03a4	W	0x00000000	SoC control register 15
GRF_SOC_CON16	0x03a8	W	0x00000000	SoC control register 16

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

### 7.3.2 Detail Register Description

#### GRF\_GPIO1D\_IOMUX

Address: Operational Base + offset (0x000c)

GPIO1D iomux control

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:7	RO	0x0	reserved
6	RW	0x0	gpio1d3_sel GPIO1D[3] iomux select 1'b0: gpio 1'b1: lcdc0_dclk
5	RO	0x0	reserved
4	RW	0x0	gpio1d2_sel GPIO1D[2] iomux select 1'b0: gpio 1'b1: lcdc0_den
3	RO	0x0	reserved
2	RW	0x0	gpio1d1_sel GPIO1D[1] iomux select 1'b0: gpio 1'b1: lcdc0_vsync

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RO	0x0	reserved
0	RW	0x0	gpio1d0_sel GPIO1D[0] iomux select 1'b0: gpio 1'b1: lcdc0_hsync

**GRF\_GPIO2A\_IOMUX**

Address: Operational Base + offset (0x0010)

GPIO2A iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RW	0x0	gpio2a7_sel GPIO2A[7] iomux select 2'b00: gpio 2'b01: cif_data9 2'b10: host_din5 2'b11: hsadc_data7
13:12	RW	0x0	gpio2a6_sel GPIO2A[6] iomux select 2'b00: gpio 2'b01: cif_data8 2'b10: host_din4 2'b11: hsadc_data6
11:10	RW	0x0	gpio2a5_sel GPIO2A[5] iomux select 2'b00: gpio 2'b01: cif_data7 2'b10: host_ckinn 2'b11: hsadc_data5
9:8	RW	0x0	gpio2a4_sel GPIO2A[4] iomux select 2'b00: gpio 2'b01: cif_data6 2'b10: host_ckinp 2'b11: hsadc_data4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:6	RW	0x0	gpio2a3_sel GPIO2A[3] iomux select 2'b00: gpio 2'b01: cif_data5 2'b10: host_din3 2'b11: hsadc_data3
5:4	RW	0x0	gpio2a2_sel GPIO2A[2] iomux select 2'b00: gpio 2'b01: cif_data4 2'b10: host_din2 2'b11: hsadc_data2
3:2	RW	0x0	gpio2a1_sel GPIO2A[1] iomux select 2'b00: gpio 2'b01: cif_data3 2'b10: host_din1 2'b11: hsadc_data1
1:0	RW	0x0	gpio2a0_sel GPIO2A[0] iomux select 2'b00: gpio 2'b01: cif_data2 2'b10: host_din0 2'b11: hsadc_data0

**GRF\_GPIO2B\_IOMUX**

Address: Operational Base + offset (0x0014)

GPIO2B iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14	RW	0x0	gpio2b7_sel GPIO2B[7] iomux select 1'b0: gpio 1'b1: cif_data11
13	RO	0x0	reserved
12	RW	0x0	gpio2b6_sel GPIO2B[6] iomux select 1'b0: gpio 1'b1: cif_data10
11	RO	0x0	reserved
10	RW	0x0	gpio2b5_sel GPIO2B[5] iomux select 1'b0: gpio 1'b1: cif_data1
9	RO	0x0	reserved
8	RW	0x0	gpio2b4_sel GPIO2B[4] iomux select 1'b0: gpio 1'b1: cif_data0
7:6	RW	0x0	gpio2b3_sel GPIO2B[3] iomux select 2'b00: gpio 2'b01: cif_clkout 2'b10: host_wkreq 2'b11: hsadcts_fail
5:4	RW	0x0	gpio2b2_sel GPIO2B[2] iomux select 2'b00: gpio 2'b01: cif_clkin 2'b10: host_wkack 2'b11: gps_clk (when hsadc_clkout_en==0) hsadc_clkout (when hsadc_clkout_en==1)
3:2	RW	0x0	gpio2b1_sel GPIO2B[1] iomux select 2'b00: gpio 2'b01: cif_href 2'b10: host_din7 2'b11: hsadcts_valid
1:0	RW	0x0	gpio2b0_sel GPIO2B[0] iomux select 2'b00: gpio 2'b01: cif_vsync 2'b10: host_din6 2'b11: hsadcts_sync

**GRF\_GPIO2C\_IOMUX**

Address: Operational Base + offset (0x0018)

GPIO2C iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:3	RO	0x0	reserved
2	RW	0x0	gpio2c1_sel GPIO2C[1] iomux select 1'b0: gpio 1'b1: i2c3cam_sda
1	RO	0x0	reserved
0	RW	0x0	gpio2c0_sel GPIO2C[0] iomux select 1'b0: gpio 1'b1: i2c3cam_scl

**GRF\_GPIO3A\_IOMUX**

Address: Operational Base + offset (0x0020)

GPIO3A iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RW	0x0	gpio3a7_sel GPIO3A[7] iomux select 2'b00: gpio 2'b01: flash0_data7 2'b10: emmc_data7 2'b11: reserved
13:12	RW	0x0	gpio3a6_sel GPIO3A[6] iomux select 2'b00: gpio 2'b01: flash0_data6 2'b10: emmc_data6 2'b11: reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:10	RW	0x0	gpio3a5_sel GPIO3A[5] iomux select 2'b00: gpio 2'b01: flash0_data5 2'b10: emmc_data5 2'b11: reserved
9:8	RW	0x0	gpio3a4_sel GPIO3A[4] iomux select 2'b00: gpio 2'b01: flash0_data4 2'b10: emmc_data4 2'b11: reserved
7:6	RW	0x0	gpio3a3_sel GPIO3A[3] iomux select 2'b00: gpio 2'b01: flash0_data3 2'b10: emmc_data3 2'b11: reserved
5:4	RW	0x0	gpio3a2_sel GPIO3A[2] iomux select 2'b00: gpio 2'b01: flash0_data2 2'b10: emmc_data2 2'b11: reserved
3:2	RW	0x0	gpio3a1_sel GPIO3A[1] iomux select 2'b00: gpio 2'b01: flash0_data1 2'b10: emmc_data1 2'b11: reserved
1:0	RW	0x0	gpio3a0_sel GPIO3A[0] iomux select 2'b00: gpio 2'b01: flash0_data0 2'b10: emmc_data0 2'b11: reserved

**GRF\_GPIO3B\_IOMUX**

Address: Operational Base + offset (0x0024)

GPIO3B iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RO	0x0	reserved
14	RW	0x0	gpio3b7_sel GPIO3B[7] iomux select 1'b0: gpio 1'b1: flash0_csn1
13	RO	0x0	reserved
12	RW	0x0	gpio3b6_sel GPIO3B[6] iomux select 1'b0: gpio 1'b1: flash0_csn0
11	RO	0x0	reserved
10	RW	0x0	gpio3b5_sel GPIO3B[5] iomux select 1'b0: gpio 1'b1: flash0_wrn
9	RO	0x0	reserved
8	RW	0x0	gpio3b4_sel GPIO3B[4] iomux select 1'b0: gpio 1'b1: flash0_cle
7	RO	0x0	reserved
6	RW	0x0	gpio3b3_sel GPIO3B[3] iomux select 1'b0: gpio 1'b1: flash0_ale
5	RO	0x0	reserved
4	RW	0x0	gpio3b2_sel GPIO3B[2] iomux select 1'b0: gpio 1'b1: flash0_rdn
3:2	RW	0x0	gpio3b1_sel GPIO3B[1] iomux select 2'b00: gpio 2'b01: flash0_wp 2'b10: emmc_pwren 2'b11: reserved
1	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	gpio3b0_sel GPIO3B[0] iomux select 1'b0: gpio 1'b1: flash0_rdy

**GRF\_GPIO3C\_IOMUX**

Address: Operational Base + offset (0x0028)

GPIO3C iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:6	RO	0x0	reserved
5:4	RW	0x0	gpio3c2_sel GPIO3C[2] iomux select 2'b00: gpio 2'b01: flash0_dqs 2'b10: emmc_clkout 2'b11: reserved
3:2	RW	0x0	gpio3c1_sel GPIO3C[1] iomux select 2'b00: gpio 2'b01: flash0_csn3 2'b10: emmc_rstnout 2'b11: reserved
1:0	RW	0x0	gpio3c0_sel GPIO3C[0] iomux select 2'b00: gpio 2'b01: flash0_csn2 2'b10: emmc_cmd 2'b11: reserved

**GRF\_GPIO3DL\_IOMUX**

Address: Operational Base + offset (0x002c)

GPIO3D iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RO	0x0	reserved
14:12	RW	0x0	gpio3d3_sel GPIO3D[3] iomux select 3'b000: gpio 3'b001: flash1_data3 3'b010: host_dout3 3'b011: mac_rxd3 3'b100: sdio1_data3 other: reserved
11	RO	0x0	reserved
10:8	RW	0x0	gpio3d2_sel GPIO3D[2] iomux select 3'b000: gpio 3'b001: flash1_data2 3'b010: host_dout2 3'b011: mac_rxd2 3'b100: sdio1_data2 other: reserved
7	RO	0x0	reserved
6:4	RW	0x0	gpio3d1_sel GPIO3D[1] iomux select 3'b000: gpio 3'b001: flash1_data1 3'b010: host_dout1 3'b011: mac_txd3 3'b100: sdio1_data1 other: reserved
3	RO	0x0	reserved
2:0	RW	0x0	gpio3d0_sel GPIO3D[0] iomux select 3'b000: gpio 3'b001: flash1_data0 3'b010: host_dout0 3'b011: mac_txd2 3'b100: sdio1_data0 other: reserved

**GRF\_GPIO3DH\_IOMUX**

Address: Operational Base + offset (0x0030)

GPIO3D iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RO	0x0	reserved
14:12	RW	0x0	gpio3d7_sel GPIO3D[7] iomux select 3'b000: gpio 3'b001: flash1_data7 3'b010: host_dout7 3'b011: mac_rxd1 3'b100: sdio1_intn other: reserved
11	RO	0x0	reserved
10:8	RW	0x0	gpio3d6_sel GPIO3D[6] iomux select 3'b000: gpio 3'b001: flash1_data6 3'b010: host_dout6 3'b011: mac_rxd0 3'b100: sdio1_bkpwr other: reserved
7	RO	0x0	reserved
6:4	RW	0x0	gpio3d5_sel GPIO3D[5] iomux select 3'b000: gpio 3'b001: flash1_data5 3'b010: host_dout5 3'b011: mac_txd1 3'b100: sdio1_wrprt other: reserved
3	RO	0x0	reserved
2:0	RW	0x0	gpio3d4_sel GPIO3D[4] iomux select 3'b000: gpio 3'b001: flash1_data4 3'b010: host_dout4 3'b011: mac_txd0 3'b100: sdio1_detectn other: reserved

**GRF\_GPIO4AL\_IOMUX**

Address: Operational Base + offset (0x0034)

## GPIO4A iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RO	0x0	reserved
14:12	RW	0x0	gpio4a3_sel GPIO4A[3] iomux select 3'b001: flash1_ale 3'b010: host_dout9 3'b011: mac_clk 3'b100: flash0_csn6 other: reserved
11	RO	0x0	reserved
10:8	RW	0x0	gpio4a2_sel GPIO4A[2] iomux select 3'b000: gpio 3'b001: flash1_rdn 3'b010: host_dout8 3'b011: mac_rxer 3'b100: flash0_csn5 other: reserved
7	RO	0x0	reserved
6:4	RW	0x0	gpio4a1_sel GPIO4A[1] iomux select 3'b000: gpio 3'b001: flash1_wp 3'b010: host_ckoutn 3'b011: mac_rxdv 3'b100: flash0_csn4 other: reserved
3:2	RO	0x0	reserved
1:0	RW	0x0	gpio4a0_sel GPIO4A[0] iomux select 2'b00: gpio 2'b01: flash1_rdy 2'b10: host_ckoutp 2'b11: mac_mdc

**GRF\_GPIO4AH\_IOMUX**

Address: Operational Base + offset (0x0038)

GPIO4A iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RO	0x0	reserved
14:12	RW	0x0	gpio4a7_sel GPIO4A[7] iomux select 3'b000: gpio 3'b001: flash1_csn1 3'b010: host_dout13 3'b011: mac_crs 3'b100: sdio1_clkout other: reserved
11	RO	0x0	reserved
10:8	RW	0x0	gpio4a6_sel GPIO4A[6] iomux select 3'b000: gpio 3'b001: flash1_csn0 3'b010: host_dout12 3'b011: mac_rxclk 3'b100: sdio1_cmd other: reserved
7:6	RO	0x0	reserved
5:4	RW	0x0	gpio4a5_sel GPIO4A[5] iomux select 2'b00: gpio 2'b01: flash1_wrn 2'b10: host_dout11 2'b11: mac_mdio
3	RO	0x0	reserved
2:0	RW	0x0	gpio4a4_sel GPIO4A[4] iomux select 3'b000: gpio 3'b001: flash1_cle 3'b010: host_dout10 3'b011: mac_txen 3'b100: flash0_csn7 other: reserved

**GRF\_GPIO4BL\_IOMUX**

Address: Operational Base + offset (0x003c)

GPIO4B iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:7	RO	0x0	reserved
6:4	RW	0x0	gpio4b1_sel GPIO4B[1] iomux select 3'b000: gpio 3'b001: flash1_csn2 3'b010: host_dout15 3'b011: mac_txclk 3'b100: sdio1_pwren other: reserved
3	RO	0x0	reserved
2:0	RW	0x0	gpio4b0_sel GPIO4B[0] iomux select 3'b000: gpio 3'b001: flash1_dqs 3'b010: host_dout14 3'b011: mac_col 3'b100: flash1_csn3 other: reserved

**GRF\_GPIO4C\_IOMUX**

Address: Operational Base + offset (0x0044)

GPIO4C iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RO	0x0	reserved
14	RW	0x0	gpio4c7_sel GPIO4C[7] iomux select 1'b0: gpio 1'b1: sdio0_data3
13	RO	0x0	reserved
12	RW	0x0	gpio4c6_sel GPIO4C[6] iomux select 1'b0: gpio 1'b1: sdio0_data2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RO	0x0	reserved
10	RW	0x0	gpio4c5_sel GPIO4C[5] iomux select 1'b0: gpio 1'b1: sdio0_data1
9	RO	0x0	reserved
8	RW	0x0	gpio4c4_sel GPIO4C[4] iomux select 1'b0: gpio 1'b1: sdio0_data0
7	RO	0x0	reserved
6	RW	0x0	gpio4c3_sel GPIO4C[3] iomux select 1'b0: gpio 1'b1: uart0bt_rtsn
5	RO	0x0	reserved
4	RW	0x0	gpio4c2_sel GPIO4C[2] iomux select 1'b0: gpio 1'b1: uart0bt_ctsn
3	RO	0x0	reserved
2	RW	0x0	gpio4c1_sel GPIO4C[1] iomux select 1'b0: gpio 1'b1: uart0bt_sout
1	RO	0x0	reserved
0	RW	0x0	gpio4c0_sel GPIO4C[0] iomux select 1'b0: gpio 1'b1: uart0bt_sin

**GRF\_GPIO4D\_IOMUX**

Address: Operational Base + offset (0x0048)

GPIO4D iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:13	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x0	gpio4d6_sel GPIO4D[6] iomux select 1'b0: gpio 1'b1: sdio0_intn
11	RO	0x0	reserved
10	RW	0x0	gpio4d5_sel GPIO4D[5] iomux select 1'b0: gpio 1'b1: sdio0_bkpwr
9	RO	0x0	reserved
8	RW	0x0	gpio4d4_sel GPIO4D[4] iomux select 1'b0: gpio 1'b1: sdio0_pwren
7	RO	0x0	reserved
6	RW	0x0	gpio4d3_sel GPIO4D[3] iomux select 1'b0: gpio 1'b1: sdio0_wrprt
5	RO	0x0	reserved
4	RW	0x0	gpio4d2_sel GPIO4D[2] iomux select 1'b0: gpio 1'b1: sdio0_detectn
3	RO	0x0	reserved
2	RW	0x0	gpio4d1_sel GPIO4D[1] iomux select 1'b0: gpio 1'b1: sdio0_clkout
1	RO	0x0	reserved
0	RW	0x0	gpio4d0_sel GPIO4D[0] iomux select 1'b0: gpio 1'b1: sdio0_cmd

**GRF\_GPIO5B\_IOMUX**

Address: Operational Base + offset (0x0050)

GPIO5B iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:14	RW	0x0	gpio5b7_sel GPIO5B[7] iomux select 2'b00: gpio 2'b01: spi0_rxd 2'b10: ts0_data7 2'b11: uart4exp_sin
13:12	RW	0x0	gpio5b6_sel GPIO5B[6] iomux select 2'b00: gpio 2'b01: spi0_txd 2'b10: ts0_data6 2'b11: uart4exp_sout
11:10	RW	0x0	gpio5b5_sel GPIO5B[5] iomux select 2'b00: gpio 2'b01: spi0_csn0 2'b10: ts0_data5 2'b11: uart4exp_rtsn
9:8	RW	0x0	gpio5b4_sel GPIO5B[4] iomux select 2'b00: gpio 2'b01: spi0_clk 2'b10: ts0_data4 2'b11: uart4exp_ctsn
7:6	RW	0x0	gpio5b3_sel GPIO5B[3] iomux select 2'b00: gpio 2'b01: uart1bb_rtsn 2'b10: ts0_data3 2'b11: reserved
5:4	RW	0x0	gpio5b2_sel GPIO5B[2] iomux select 2'b00: gpio 2'b01: uart1bb_ctsn 2'b10: ts0_data2 2'b11: reserved
3:2	RW	0x0	gpio5b1_sel GPIO5B[1] iomux select 2'b00: gpio 2'b01: uart1bb_sout 2'b10: ts0_data1 2'b11: reserved

Bit	Attr	Reset Value	Description
1:0	RW	0x0	gpio5b0_sel GPIO5B[0] iomux select 2'b00: gpio 2'b01: uart1bb_sin 2'b10: ts0_data0 2'b11: reserved

**GRF\_GPIO5C\_IOMUX**

Address: Operational Base + offset (0x0054)

GPIO5C iomux control

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:7	RO	0x0	reserved
6	RW	0x0	gpio5c3_sel GPIO5C[3] iomux select 1'b0: gpio 1'b1: ts0_err
5	RO	0x0	reserved
4	RW	0x0	gpio5c2_sel GPIO5C[2] iomux select 1'b0: gpio 1'b1: ts0_clk
3	RO	0x0	reserved
2	RW	0x0	gpio5c1_sel GPIO5C[1] iomux select 1'b0: gpio 1'b1: ts0_valid
1:0	RW	0x0	gpio5c0_sel GPIO5C[0] iomux select 2'b00: gpio 2'b01: spi0_csn1 2'b10: ts0_sync 2'b11: reserved

**GRF\_GPIO6A\_IOMUX**

Address: Operational Base + offset (0x005c)

GPIO6A iomux control

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RO	0x0	reserved
14	RW	0x0	gpio6a7_sel GPIO6A[7] iomux select 1'b0: gpio 1'b1: i2s_sdo3
13	RO	0x0	reserved
12	RW	0x0	gpio6a6_sel GPIO6A[6] iomux select 1'b0: gpio 1'b1: i2s_sdo2
11	RO	0x0	reserved
10	RW	0x0	gpio6a5_sel GPIO6A[5] iomux select 1'b0: gpio 1'b1: i2s_sdo1
9	RO	0x0	reserved
8	RW	0x0	gpio6a4_sel GPIO6A[4] iomux select 1'b0: gpio 1'b1: i2s_sdo0
7	RO	0x0	reserved
6	RW	0x0	gpio6a3_sel GPIO6A[3] iomux select 1'b0: gpio 1'b1: i2s_sdi
5	RO	0x0	reserved
4	RW	0x0	gpio6a2_sel GPIO6A[2] iomux select 1'b0: gpio 1'b1: i2s_lrcktx
3	RO	0x0	reserved
2	RW	0x0	gpio6a1_sel GPIO6A[1] iomux select 1'b0: gpio 1'b1: i2s_lrckrx
1	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	gpio6a0_sel GPIO6A[0] iomux select 1'b0: gpio 1'b1: i2s_sclk

**GRF\_GPIO6B\_IOMUX**

Address: Operational Base + offset (0x0060)

GPIO6B iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:7	RO	0x0	reserved
6	RW	0x0	gpio6b3_sel GPIO6B[3] iomux select 1'b0: gpio 1'b1: spdif_tx
5	RO	0x0	reserved
4	RW	0x0	gpio6b2_sel GPIO6B[2] iomux select 1'b0: gpio 1'b1: i2c1audio_scl
3	RO	0x0	reserved
2	RW	0x0	gpio6b1_sel GPIO6B[1] iomux select 1'b0: gpio 1'b1: i2c1audio_sda
1	RO	0x0	reserved
0	RW	0x0	gpio6b0_sel GPIO6B[0] iomux select 1'b0: gpio 1'b1: i2s_clk

**GRF\_GPIO6C\_IOMUX**

Address: Operational Base + offset (0x0064)

GPIO6C iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:13	RO	0x0	reserved
12	RW	0x1	gpio6c6_sel GPIO6C[6] iomux select 1'b0: gpio 1'b1: sdmmc0_dectn
11	RO	0x0	reserved
10	RW	0x1	gpio6c5_sel GPIO6C[5] iomux select 1'b0: gpio 1'b1: sdmmc0_cmd
9:8	RW	0x1	gpio6c4_sel GPIO6C[4] iomux select 2'b00: gpio 2'b01: sdmmc0_clkout 2'b10: jtag_tdo 2'b11: reserved
7:6	RW	0x1	gpio6c3_sel GPIO6C[3] iomux select 2'b00: gpio 2'b01: sdmmc0_data3 2'b10: jtag_tck 2'b11: reserved
5:4	RW	0x1	gpio6c2_sel GPIO6C[2] iomux select 2'b00: gpio 2'b01: sdmmc0_data2 2'b10: jtag_tdi 2'b11: reserved
3:2	RW	0x1	gpio6c1_sel GPIO6C[1] iomux select 2'b00: gpio 2'b01: sdmmc0_data1 2'b10: jtag_trstn 2'b11: reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RW	0x1	gpio6c0_sel GPIO6C[0] iomux select 2'b00: gpio 2'b01: sdmmc0_data0 2'b10: jtag_tms 2'b11: reserved

**GRF\_GPIO7A\_IOMUX**

Address: Operational Base + offset (0x006c)

GPIO7A iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RW	0x0	gpio7a7_sel GPIO7A[7] iomux select 2'b00: gpio 2'b01: uart3gps_sin 2'b10: gps_mag 2'b11: hsadct1_data0
13:3	RO	0x0	reserved
2	RW	0x0	gpio7a1_sel GPIO7A[1] iomux select 1'b0: gpio 1'b1: pwm_1
1:0	RW	0x0	gpio7a0_sel GPIO7A[0] iomux select 2'b00: gpio 2'b01: pwm_0 2'b10: vop0_pwm 2'b11: vop1_pwm

**GRF\_GPIO7B\_IOMUX**

Address: Operational Base + offset (0x0070)

GPIO7B iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:14	RW	0x0	gpio7b7_sel GPIO7B[7] iomux select 2'b00: gpio 2'b01: isp_shuttertrig 2'b10: spi1_txd 2'b11: reserved
13:12	RW	0x0	gpio7b6_sel GPIO7B[6] iomux select 2'b00: gpio 2'b01: isp_prelighttrig 2'b10: spi1_rxd 2'b11: reserved
11:10	RW	0x0	gpio7b5_sel GPIO7B[5] iomux select 2'b00: gpio 2'b01: isp_flashtrigout 2'b10: spi1_csn0 2'b11: reserved
9:8	RW	0x0	gpio7b4_sel GPIO7B[4] iomux select 2'b00: gpio 2'b01: isp_shutteren 2'b10: spi1_clk 2'b11: reserved
7:6	RW	0x0	gpio7b3_sel GPIO7B[3] iomux select 2'b00: gpio 2'b01: usb_drvvbus1 2'b10: edp_hotplug 2'b11: reserved
5:4	RW	0x0	gpio7b2_sel GPIO7B[2] iomux select 2'b00: gpio 2'b01: uart3gps_rtsn 2'b10: usb_drvvbus0 2'b11: reserved
3:2	RW	0x0	gpio7b1_sel GPIO7B[1] iomux select 2'b00: gpio 2'b01: uart3gps_ctsn 2'b10: gps_rfclk 2'b11: gpst1_clk

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RW	0x0	gpio7b0_sel GPIO7B[0] iomux select 2'b00: gpio 2'b01: uart3gps_sout 2'b10: gps_sig 2'b11: hsadct1_data1

**GRF\_GPIO7CL\_IOMUX**

Address: Operational Base + offset (0x0074)

GPIO7CL iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved
13:12	RW	0x0	gpio7c3_sel GPIO7C[3] iomux select 2'b00: gpio 2'b01: i2c5hdmi_sda 2'b10: edphdmi2c_sda 2'b11: reserved
11:9	RO	0x0	reserved
8	RW	0x0	gpio7c2_sel GPIO7C[2] iomux select 1'b0: gpio 1'b1: i2c4tp_scl
7:5	RO	0x0	reserved
4	RW	0x0	gpio7c1_sel GPIO7C[1] iomux select 1'b0: gpio 1'b1: i2c4tp_sda
3:2	RO	0x0	reserved
1:0	RW	0x0	gpio7c0_sel GPIO7C[0] iomux select 2'b00: gpio 2'b01: isp_flashtrigin 2'b10: edphdmi_cecinoutt1 2'b11: reserved

**GRF\_GPIO7CH\_IOMUX**

Address: Operational Base + offset (0x0078)

GPIO7CH iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RO	0x0	reserved
14:12	RW	0x0	gpio7c7_sel GPIO7C[7] iomux select 3'b000: gpio 3'b001: uart2dbg_sout 3'b010: uart2dbg_sirout 3'b011: pwm_3 3'b100: edphdmi_cecinout other: reserved
11:10	RO	0x0	reserved
9:8	RW	0x0	gpio7c6_sel GPIO7C[6] iomux select 2'b00: gpio 2'b01: uart2dbg_sin 2'b10: uart2dbg_sirin 2'b11: pwm_2
7:2	RO	0x0	reserved
1:0	RW	0x0	gpio7c4_sel GPIO7C[4] iomux select 2'b00: gpio 2'b01: i2c5hdmi_scl 2'b10: edphdmi2c_scl 2'b11: reserved

**GRF\_GPIO8A\_IOMUX**

Address: Operational Base + offset (0x0080)

GPIO8A iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:14	RW	0x0	gpio8a7_sel GPIO8A[7] iomux select 2'b00: gpio 2'b01: spi2_csn0 2'b10: sc_detect 2'b11: reserve
13:12	RW	0x0	gpio8a6_sel GPIO8A[6] iomux select 2'b00: gpio 2'b01: spi2_clk 2'b10: sc_io 2'b11: reserve
11:10	RW	0x0	gpio8a5_sel GPIO8A[5] iomux select 2'b00: gpio 2'b01: i2c2sensor_scl 2'b10: sc_clk 2'b11: reserved
9:8	RW	0x0	gpio8a4_sel GPIO8A[4] iomux select 2'b00: gpio 2'b01: i2c2sensor_sda 2'b10: sc_rst 2'b11: reserved
7:6	RW	0x0	gpio8a3_sel GPIO8A[3] iomux select 2'b00: gpio 2'b01: spi2_csn1 2'b10: sc_iot1 2'b11: reserved
5	RO	0x0	reserved
4	RW	0x0	gpio8a2_sel GPIO8A[2] iomux select 1'b0: gpio 1'b1: sc_detectt1
3:2	RW	0x0	gpio8a1_sel GPIO8A[1] iomux select 2'b00: gpio 2'b01: ps2_data 2'b10: sc_vcc33v 2'b11: reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RW	0x0	gpio8a0_sel GPIO8A[0] iomux select 2'b00: gpio 2'b01: ps2_clk 2'b10: sc_vcc18v 2'b11: reserved

**GRF\_GPIO8B\_IOMUX**

Address: Operational Base + offset (0x0084)

GPIO8B iomux control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:4	RO	0x0	reserved
3:2	RW	0x0	gpio8b1_sel GPIO8B[1] iomux select 2'b00: gpio 2'b01: spi2_txd 2'b10: sc_clk 2'b11: reserve
1:0	RW	0x0	gpio8b0_sel GPIO8B[0] iomux select 2'b00: gpio 2'b01: spi2_rxd 2'b10: sc_rst 2'b11: reserve

**GRF\_GPIO1H\_SR**

Address: Operational Base + offset (0x0104)

GPIO1C/D SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:8	RW	0x0f	<p>gpio1d_sr</p> <p>GPIO1D slew rate control for each bit</p> <p>1'b0: slow (half frequency)</p> <p>1'b1: fast</p>
7:0	RO	0x0	reserved

**GRF\_GPIO2L\_SR**

Address: Operational Base + offset (0x0108)

GPIO2A/B SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:8	RW	0x00	<p>gpio2b_sr</p> <p>GPIO2B slew rate control for each bit</p> <p>1'b0: slow (half frequency)</p> <p>1'b1: fast</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	RW	0x00	gpio2a_sr GPIO2A slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast

**GRF\_GPIO2H\_SR**

Address: Operational Base + offset (0x010c)

GPIO2C/D SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:8	RO	0x0	reserved
7:0	RW	0x00	gpio2c_sr GPIO2C slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast

**GRF\_GPIO3L\_SR**

Address: Operational Base + offset (0x0110)

GPIO3A/B SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:8	RW	0x20	<p>gpio3b_sr</p> <p>GPIO3B slew rate control for each bit</p> <p>1'b0: slow (half frequency)</p> <p>1'b1: fast</p>
7:0	RW	0xff	<p>gpio3a_sr</p> <p>GPIO3A slew rate control for each bit</p> <p>1'b0: slow (half frequency)</p> <p>1'b1: fast</p>

**GRF\_GPIO3H\_SR**

Address: Operational Base + offset (0x0114)

GPIO3C/D SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:8	RW	0xff	gpio3d_sr GPIO3D slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast
7:0	RW	0x04	gpio3c_sr GPIO3C slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast

**GRF\_GPIO4L\_SR**

Address: Operational Base + offset (0x0118)

GPIO4A/B SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:8	RW	0x01	gpio4b_sr GPIO4B slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast
7:0	RW	0x20	gpio4a_sr GPIO4A slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast

**GRF\_GPIO4H\_SR**

Address: Operational Base + offset (0x011c)

GPIO4C/D SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:8	RW	0x00	<p>gpio4d_sr</p> <p>GPIO4D slew rate control for each bit</p> <p>1'b0: slow (half frequency)</p> <p>1'b1: fast</p>
7:0	RW	0x00	<p>gpio4c_sr</p> <p>GPIO4C slew rate control for each bit</p> <p>1'b0: slow (half frequency)</p> <p>1'b1: fast</p>

**GRF\_GPIO5L\_SR**

Address: Operational Base + offset (0x0120)

GPIO5A/B SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:8	RW	0x00	gpio5b_sr GPIO5B slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast
7:0	RO	0x0	reserved

**GRF\_GPIO5H\_SR**

Address: Operational Base + offset (0x0124)

GPIO5C/D SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:8	RO	0x0	reserved
7:0	RW	0x00	gpio5c_sr GPIO5C slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast

**GRF\_GPIO6L\_SR**

Address: Operational Base + offset (0x0128)

GPIO6A/B SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:8	RW	0x01	<p>gpio6b_sr</p> <p>GPIO6B slew rate control for each bit</p> <p>1'b0: slow (half frequency)</p> <p>1'b1: fast</p>
7:0	RW	0x00	<p>gpio6a_sr</p> <p>GPIO6A slew rate control for each bit</p> <p>1'b0: slow (half frequency)</p> <p>1'b1: fast</p>

**GRF\_GPIO6H\_SR**

Address: Operational Base + offset (0x012c)

GPIO6C/D SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:8	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	RW	0x10	gpio6c_sr GPIO6C slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast

**GRF\_GPIO7L\_SR**

Address: Operational Base + offset (0x0130)

GPIO7A/B SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:8	RW	0x00	gpio7b_sr GPIO7B slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast
7:0	RW	0x00	gpio7a_sr GPIO7A slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast

**GRF\_GPIO7H\_SR**

Address: Operational Base + offset (0x0134)

GPIO7C/D SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:8	RO	0x0	reserved
7:0	RW	0x00	<p>gpio7c_sr</p> <p>GPIO7C slew rate control for each bit</p> <p>1'b0: slow (half frequency)</p> <p>1'b1: fast</p>

**GRF\_GPIO8L\_SR**

Address: Operational Base + offset (0x0138)

GPIO8A/B SR control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:8	RW	0x00	<p>gpio8b_sr</p> <p>GPIO8B slew rate control for each bit</p> <p>1'b0: slow (half frequency)</p> <p>1'b1: fast</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	RW	0x00	gpio8a_sr GPIO8A slew rate control for each bit 1'b0: slow (half frequency) 1'b1: fast

**GRF\_GPIO1D\_P**

Address: Operational Base + offset (0x014c)

GPIO1D PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0xaaaa	gpio1d_p GPIO1D PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO2A\_P**

Address: Operational Base + offset (0x0150)

GPIO2A PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0xaaaa	<p>gpio2a_p</p> <p>GPIO2A PU/PD programmation section, every GPIO bit corresponding to 2bits</p> <p>2'b00: Z(Noraml operaton);</p> <p>2'b01: weak 1(pull-up);</p> <p>2'b10: weak 0(pull-down);</p> <p>2'b11: Repeater(Bus keeper)</p>

**GRF\_GPIO2B\_P**

Address: Operational Base + offset (0x0154)

GPIO2B PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0xaaaa	gpio2b_p GPIO2B PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO2C\_P**

Address: Operational Base + offset (0x0158)

GPIO2C PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0xaaa5	gpio2c_p GPIO2C PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO3A\_P**

Address: Operational Base + offset (0x0160)

GPIO3A PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5555	<p>gpio3a_p</p> <p>GPIO3A PU/PD programmation section, every GPIO bit corresponding to 2bits</p> <p>2'b00: Z(Noraml operaton);</p> <p>2'b01: weak 1(pull-up);</p> <p>2'b10: weak 0(pull-down);</p> <p>2'b11: Repeater(Bus keeper)</p>

**GRF\_GPIO3B\_P**

Address: Operational Base + offset (0x0164)

GPIO3B PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x5699	gpio3b_p GPIO3B PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO3C\_P**

Address: Operational Base + offset (0x0168)

GPIO3C PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0xaaaa5	gpio3c_p GPIO3C PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO3D\_P**

Address: Operational Base + offset (0x016c)

GPIO3D PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5555	<p>gpio3d_p</p> <p>GPIO3D PU/PD programmation section, every GPIO bit corresponding to 2bits</p> <p>2'b00: Z(Noraml operaton);</p> <p>2'b01: weak 1(pull-up);</p> <p>2'b10: weak 0(pull-down);</p> <p>2'b11: Repeater(Bus keeper)</p>

**GRF\_GPIO4A\_P**

Address: Operational Base + offset (0x0170)

GPIO4A PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x5555	gpio4a_p GPIO4A PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO4B\_P**

Address: Operational Base + offset (0x0174)

GPIO4B PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0xaaaa5	gpio4b_p GPIO4B PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO4C\_P**

Address: Operational Base + offset (0x0178)

GPIO4C PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5559	<p>gpio4c_p</p> <p>GPIO4C PU/PD programmation section, every GPIO bit corresponding to 2bits</p> <p>2'b00: Z(Noraml operaton);</p> <p>2'b01: weak 1(pull-up);</p> <p>2'b10: weak 0(pull-down);</p> <p>2'b11: Repeater(Bus keeper)</p>

**GRF\_GPIO4D\_P**

Address: Operational Base + offset (0x017c)

GPIO4D PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x5a99	gpio4d_p GPIO4D PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO5B\_P**

Address: Operational Base + offset (0x0184)

GPIO5B PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0x6559	gpio5b_p GPIO5B PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO5C\_P**

Address: Operational Base + offset (0x0188)

GPIO5C PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0xaaa9	<p>gpio5c_p</p> <p>GPIO5C PU/PD programmation section, every GPIO bit corresponding to 2bits</p> <p>2'b00: Z(Noraml operaton);</p> <p>2'b01: weak 1(pull-up);</p> <p>2'b10: weak 0(pull-down);</p> <p>2'b11: Repeater(Bus keeper)</p>

**GRF\_GPIO6A\_P**

Address: Operational Base + offset (0x0190)

GPIO6A PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0xaaaa	gpio6a_p GPIO6A PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO6B\_P**

Address: Operational Base + offset (0x0194)

GPIO6B PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0xaa96	gpio6b_p GPIO6B PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO6C\_P**

Address: Operational Base + offset (0x0198)

GPIO6C PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5655	<p>gpio6c_p</p> <p>GPIO6C PU/PD programmation section, every GPIO bit corresponding to 2bits</p> <p>2'b00: Z(Noraml operaton);</p> <p>2'b01: weak 1(pull-up);</p> <p>2'b10: weak 0(pull-down);</p> <p>2'b11: Repeater(Bus keeper)</p>

**GRF\_GPIO7A\_P**

Address: Operational Base + offset (0x01a0)

GPIO7A PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x59aa	gpio7a_p GPIO7A PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO7B\_P**

Address: Operational Base + offset (0x01a4)

GPIO7B PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0xa696	gpio7b_p GPIO7B PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO7C\_P**

Address: Operational Base + offset (0x01a8)

GPIO7C PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5955	<p>gpio7c_p</p> <p>GPIO7C PU/PD programmation section, every GPIO bit corresponding to 2bits</p> <p>2'b00: Z(Noraml operaton);</p> <p>2'b01: weak 1(pull-up);</p> <p>2'b10: weak 0(pull-down);</p> <p>2'b11: Repeater(Bus keeper)</p>

**GRF\_GPIO8A\_P**

Address: Operational Base + offset (0x01b0)

GPIO8A PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x6555	gpio8a_p GPIO8A PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO8B\_P**

Address: Operational Base + offset (0x01b4)

GPIO8B PU/PD control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0xaaaa	gpio8b_p GPIO8B PU/PD programmation section, every GPIO bit corresponding to 2bits 2'b00: Z(Noraml operaton); 2'b01: weak 1(pull-up); 2'b10: weak 0(pull-down); 2'b11: Repeater(Bus keeper)

**GRF\_GPIO1D\_E**

Address: Operational Base + offset (0x01cc)

GPIO1D drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x55aa	<p>gpio1d_e</p> <p>GPIO1D drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA</p> <p>2'b01: 4mA</p> <p>2'b10: 8mA</p> <p>2'b11: 12mA</p>

**GRF\_GPIO2A\_E**

Address: Operational Base + offset (0x01d0)

GPIO2A drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0xaaaa	gpio2a_e GPIO2A drive strength control, every GPIO bit corresponding to 2bits 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA

**GRF\_GPIO2B\_E**

Address: Operational Base + offset (0x01d4)

GPIO2B drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0xaaaa	gpio2b_e GPIO2B drive strength control, every GPIO bit corresponding to 2bits 2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA

**GRF\_GPIO2C\_E**

Address: Operational Base + offset (0x01d8)

GPIO2C drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5555	<p>gpio2c_e</p> <p>GPIO2C drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA</p> <p>2'b01: 4mA</p> <p>2'b10: 8mA</p> <p>2'b11: 12mA</p>

**GRF\_GPIO3A\_E**

Address: Operational Base + offset (0x01e0)

GPIO3A drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0xaaaa	<p>gpio3a_e</p> <p>GPIO3A drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO3B\_E**

Address: Operational Base + offset (0x01e4)

GPIO3B drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5955	<p>gpio3b_e</p> <p>GPIO3B drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO3C\_E**

Address: Operational Base + offset (0x01e8)

GPIO3C drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5565	<p>gpio3c_e</p> <p>GPIO3C drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA</p> <p>2'b01: 4mA</p> <p>2'b10: 8mA</p> <p>2'b11: 12mA</p>

**GRF\_GPIO3D\_E**

Address: Operational Base + offset (0x01ec)

GPIO3D drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0xaaaa	<p>gpio3d_e</p> <p>GPIO3D drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO4A\_E**

Address: Operational Base + offset (0x01f0)

GPIO4A drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5955	<p>gpio4a_e</p> <p>GPIO4A drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO4B\_E**

Address: Operational Base + offset (0x01f4)

GPIO4B drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5556	<p>gpio4b_e</p> <p>GPIO4B drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA</p> <p>2'b01: 4mA</p> <p>2'b10: 8mA</p> <p>2'b11: 12mA</p>

**GRF\_GPIO4C\_E**

Address: Operational Base + offset (0x01f8)

GPIO4C drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x5555	<p>gpio4c_e</p> <p>GPIO4C drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO4D\_E**

Address: Operational Base + offset (0x01fc)

GPIO4D drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5555	<p>gpio4d_e</p> <p>GPIO4D drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO5B\_E**

Address: Operational Base + offset (0x0204)

GPIO5B drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5555	<p>gpio5b_e</p> <p>GPIO5B drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA</p> <p>2'b01: 4mA</p> <p>2'b10: 8mA</p> <p>2'b11: 12mA</p>

**GRF\_GPIO5C\_E**

Address: Operational Base + offset (0x0208)

GPIO5C drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x5555	<p>gpio5c_e</p> <p>GPIO5C drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO6A\_E**

Address: Operational Base + offset (0x0210)

GPIO6A drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5555	<p>gpio6a_e</p> <p>GPIO6A drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO6B\_E**

Address: Operational Base + offset (0x0214)

GPIO6B drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5555	<p>gpio6b_e</p> <p>GPIO6B drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA</p> <p>2'b01: 4mA</p> <p>2'b10: 8mA</p> <p>2'b11: 12mA</p>

**GRF\_GPIO6C\_E**

Address: Operational Base + offset (0x0218)

GPIO6C drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x5555	<p>gpio6c_e</p> <p>GPIO6C drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO7A\_E**

Address: Operational Base + offset (0x0220)

GPIO7A drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5555	<p>gpio7a_e</p> <p>GPIO7A drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO7B\_E**

Address: Operational Base + offset (0x0224)

GPIO7B drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5555	<p>gpio7b_e</p> <p>GPIO7B drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA</p> <p>2'b01: 4mA</p> <p>2'b10: 8mA</p> <p>2'b11: 12mA</p>

**GRF\_GPIO7C\_E**

Address: Operational Base + offset (0x0228)

GPIO7C drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x5555	<p>gpio7c_e</p> <p>GPIO7C drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO8A\_E**

Address: Operational Base + offset (0x0230)

GPIO8A drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5555	<p>gpio8a_e</p> <p>GPIO8A drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA 2'b01: 4mA 2'b10: 8mA 2'b11: 12mA</p>

**GRF\_GPIO8B\_E**

Address: Operational Base + offset (0x0234)

GPIO8B drive strength control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x5555	<p>gpio8b_e</p> <p>GPIO8B drive strength control, every GPIO bit corresponding to 2bits</p> <p>2'b00: 2mA</p> <p>2'b01: 4mA</p> <p>2'b10: 8mA</p> <p>2'b11: 12mA</p>

**GRF\_GPIO\_SMT**

Address: Operational Base + offset (0x0240)

GPIO smitter control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:12	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x1	gpio8a1_smt GPIO8A_1 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled
10	RW	0x1	gpio8a0_smt GPIO8A_0 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled
9	RW	0x1	gpio7c4_smt GPIO7C_4 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled
8	RW	0x1	gpio7c3_smt GPIO7C_3 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled
7	RW	0x1	gpio7c2_smt GPIO7C_2 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled
6	RW	0x1	gpio7c1_smt GPIO7C_1 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled
5	RW	0x1	gpio2c1_smt GPIO2C_1 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled
4	RW	0x1	gpio2c0_smt GPIO2C_0 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled
3	RW	0x1	gpio6b2_smt GPIO6B_2 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled
2	RW	0x1	gpio6b1_smt GPIO6B_1 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled
1	RW	0x1	gpio8a5_smt GPIO8A_5 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x1	gpio8a4_smt GPIO8A_4 SMT control 1'b0: No hysteresis 1'b1: Schmitt trigger enabled

**GRF\_SOC\_CON0**

Address: Operational Base + offset (0x0244)

SoC control register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RW	0x0	pause_mmc_peri PERI MMC AHB bus arbiter pause control
14	RW	0x0	pause_emem_peri PERI EMEM AHB bus arbiter pause control
13	RW	0x0	pause_usb_peri PERI USB AHB bus arbiter pause control
12	RW	0x1	grf_force_jtag Force select jtag function from sdmmc0 IO
11	RW	0x1	grf_core_idle_req_mode_sel1 core idle request mode selection 1
10	RW	0x1	grf_core_idle_req_mode_sel0 core idle request mode selection 0
9	RW	0x0	ddr1_16bit_en DDR Channel 1 interface 16bit enable
8	RW	0x0	ddr0_16bit_en DDR Channel 0 interface 16bit enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	vcodec_sel vdpu vepu clock select 1'b0: select vepu aclk as vcodec main clock 1'b1: select vdpu aclk as vcodec main clock
6	RW	0x0	upctl1_c_active_in Channel 1 DDR clock active in External signal from system that flags if a hardware low power request can be accepted or should always be denied. 1'b0: may be accepted 1'b1: will be denied
5	RW	0x0	upctl0_c_active_in Channel 0 DDR clock active in External signal from system that flags if a hardware low power request can be accepted or should always be denied. 1'b0: may be accepted 1'b1: will be denied
4	RW	0x1	msch1_mainddr3 Channel 1 DDR3 mode control 1'b1: DDR3 mode
3	RW	0x1	msch0_mainddr3 Channel 0 DDR3 mode control 1'b1: DDR3 mode
2	RW	0x0	msch1_mainpartialpop msch1_mainpartialpop bit control
1	RW	0x0	msch0_mainpartialpop msch0_mainpartialpop bit control
0	RO	0x0	reserved

**GRF\_SOC\_CON1**

Address: Operational Base + offset (0x0248)

SoC control register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15	RO	0x0	reserved
14	RW	0x1	<p>rmii_mode</p> <p>RMII mode selection</p> <p>1'b1: RMII mode</p>
13:12	RW	0x0	<p>gmac_clk_sel</p> <p>RGMII clock selection</p> <p>2'b00: 125MHz</p> <p>2'b11: 25MHz</p> <p>2'b10: 2.5MHz</p>
11	RW	0x0	<p>rmii_clk_sel</p> <p>RMII clock selection</p> <p>1'b1: 25MHz</p> <p>1'b0: 2.5MHz</p>
10	RW	0x0	<p>gmac_speed</p> <p>MAC speed</p> <p>1'b1: 100-Mbps</p> <p>1'b0: 10-Mbps</p>
9	RW	0x0	<p>gmac_flowctrl</p> <p>GMAC transmit flow control</p> <p>When set high, instructs the GMAC to transmit PAUSE Control frames in Full-duplex mode. In Half-duplex mode, the GMAC enables the Back-pressure function until this signal is made low again</p>
8:6	RW	0x1	<p>gmac_phy_intf_sel</p> <p>PHY interface select</p> <p>3'b001: RGMII</p> <p>3'b100: RMII</p> <p>All others: Reserved</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	host_remap Host interface remap control
4:0	RO	0x0	reserved

**GRF\_SOC\_CON2**

Address: Operational Base + offset (0x024c)

SoC control register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:14	RO	0x0	reserved
13	RW	0x0	upctl1_lpddr3_odt_en Channel 1 DDR odt enable in LPDDR3 mode 1'b1: ODT enable 1'b0: ODT disable
12	RW	0x0	upctl1_bst_disable Channel 1 DDR controller burst termination disable control 1'b1: disable 1'b0: enable
11	RW	0x0	lpddr3_en1 Channel 1 LPDDR3 mode control 1'b1: LPDDR3 mode
10	RW	0x0	upctl0_lpddr3_odt_en Channel 0 DDR odt enable in LPDDR3 mode 1'b1: ODT enable 1'b0: ODT disable
9	RW	0x0	upctl0_bst_disable Channel 0 DDR controller burst termination disable control 1'b1: disable 1'b0: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	RW	0x0	lpddr3_en0 Channel 0 LPDDR3 mode control 1'b1: LPDDR3 mode
7	RW	0x0	grf_poc_flash0_ctrl Flash0 IO domain 1.8V selection source 1'b0: GPIO3C_3 to decide the flash0 IO domain voltage, when GPIO3C_3 high, the voltage is 1.8V 1'b1: grf_io_vsel[2] to decide the flash0 IO domain voltage, when grf_io_vsel[2] high, the voltage is 1.8V
6	RW	0x0	simcard_mux_sel sim card iomux solution selection 1'b1: use GPIO8A[5:2] 1'b0: use GPIO8A[7:6] and GPIO8B[1:0]
5:2	RO	0x0	reserved
1	RW	0x1	grf_spdif_2ch_en SPDIF solution selection 1'b0: 8CH SPDIF 1'b1: 2CH SPDIF
0	RW	0x0	pwm_sel PWM solution selection 1'b1: RK PWM 1'b0: PWM(old)

**GRF\_SOC\_CON3**

Address: Operational Base + offset (0x0250)

SoC control register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RW	0x0	rxclk_dly_ena_gmac RGMII RX clock delayline enable 1'b1: enable 1'b0: disable
14	RW	0x0	txclk_dly_ena_gmac RGMII TX clock delayline enable 1'b1: enable 1'b0: disable
13:7	RW	0x10	clk_rx_dl_cfg_gmac RGMII RX clock delayline value
6:0	RW	0x10	clk_tx_dl_cfg_gmac RGMII TX clock delayline value

**GRF\_SOC\_CON4**

Address: Operational Base + offset (0x0254)

SoC control register 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RW	0x0	dfi_eff_stat_en1 Channel 1 DFI monitor efficiency statistics enable
14	RW	0x0	dfi_eff_stat_en0 Channel 0 DFI monitor efficiency statistics enable
13	RW	0x0	mobile_ddr_sel Mobile DDR selection in DFI monitor 1'b1: mobile DDR(LPDDR2/LPDDR3) 1'b0: DDR3
12:10	RW	0x1	host_l3_ocp_sconnect_grf Host interface l3_ocp_sconnect signal control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9:8	RW	0x2	host_txport_rst_val_grf Host interface txport_rst_val signal control
7:6	RW	0x0	host_rxport_rst_val_grf Host interface rxport_rst_val signal control
5	RW	0x0	host_wakereq_grf Host interface wakereq signal control
4:3	RW	0x0	host_eoi_in_grf Host interface eoi_in signal control
2	RW	0x1	host_mstandy_in_grf Host interface mstandy_in signal control
1	RW	0x1	host_mwait_grf Host interface mwait signal control
0	RW	0x1	host_sidle_req_grf Host interface sidle_req signal control

**GRF\_SOC\_CON5**

Address: Operational Base + offset (0x0258)

SoC control register 5

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RW	0x1	host_mux_sel Host interface mux selection 1'b1: 8bits input, 16bits output 1'b0: 8bits output, 16bits input
14	RW	0x0	tsp0_inout_sel TSP0 input/output selection 1'b1: output 1'b0: input

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13	RW	0x0	hsadc_clkout_en hsadc clkout enable 1'b1: hsadc_clkout 1'b0: gps_clk
12:3	RW	0x190	host_fclk_freq_RST_val_grf Host interface fclk_freq_RST_val signal control
2:1	RW	0x3	host_I3_iocp_mconnect_grf Host interface I3_iocp_mconnect signal control
0	RW	0x1	host_I3_ocp_mdiscbehave_grf Host interface I3_ocp_mdiscbehave signal control

**GRF\_SOC\_CON6**

Address: Operational Base + offset (0x025c)

SoC control register 6

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RW	0x1	grf_hdmi_edp_sel HDMI source selection 1'b1: from HDMI controller 1'b0: from eDP controller
14	RW	0x0	dsi_csi_testbus_sel MIPI PHY TX1RX1 test bus source selection 1'b1: CSI host 1'b0: DSI host1
13	RW	0x0	hsadc_extclk_mux_sel HSADC external clock source selection 1'b1: GPIO7B[1] 1'b0: GPIO2B[2]

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x0	clk_27m_mux_sel 27M clock input source selection 1'b1: GPIO0C[1] 1'b0: GPIO0B[5]
11	RW	0x0	grf_con_dsi1_dpicolor DSI host1 dpicolor bit control
10	RW	0x0	grf_con_dsi1_dpishutdn DSI host1 dpishutdn bit control
9	RW	0x0	grf_con_dsi1_lcdc_sel DSI host1 data from VOP selection 1'b1: VOP LIT output to DSI host1 1'b0: VOP BIG output to DSI host1
8	RW	0x0	grf_con_dsi0_dpicolor DSI host0 dpicolor bit control
7	RW	0x0	grf_con_dsi0_dpishutdn DSI host0 dpishutdn bit control
6	RW	0x0	grf_con_dsi0_lcdc_sel DSI host0 data from VOP selection 1'b1: VOP LIT output to DSI host0 1'b0: VOP BIG output to DSI host0
5	RW	0x0	grf_con_edp_lcdc_sel eDP data from VOP selection 1'b1: VOP LIT output to eDP 1'b0: VOP BIG output to eDP
4	RW	0x0	grf_con_hdmi_lcdc_sel HDMI data from VOP selection 1'b1: VOP LIT output to HDMI 1'b0: VOP BIG output to HDMI
3	RW	0x0	grf_con_lvds_lcdc_sel LVDS data from VOP selection 1'b1: VOP LIT output to LVDS 1'b0: VOP BIG output to LVDS
2	RW	0x0	grf_con_iep_vop_sel IEP connect to VOP selection 1'b1: IEP connect to VOP LIT 1'b0: IEP connect to VOP BIG
1	RW	0x0	grf_con_isp_dphy_sel ISP connect to MIPI PHY selection 1'b1: MIPI PHY TX1RX1 1'b0: MIPI PHY RX0
0	RW	0x0	grf_con_disable_isp Disable ISP control

**GRF\_SOC\_CON7**

Address: Operational Base + offset (0x0260)

## SoC control register 7

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15	RW	0x0	<p>grf_lvds_pwrdown</p> <p>LVDS PHY power down control</p> <p>1'b1: power down</p> <p>1'b0: power up</p>
14	RO	0x0	reserved
13	RW	0x0	<p>grf_lvds_lcdc_trace_sel</p> <p>LVDS IO used as trace bus enable</p> <p>1'b1: used as trace bus</p> <p>1'b0: used as LVDS IO or LCDC RGB output port</p>
12	RW	0x0	<p>grf_lvds_con_enable_2</p> <p>LVDS controller enable_2 signal control</p>
11	RW	0x0	<p>grf_lvds_con_enable_1</p> <p>LVDS controller enable_1 signal control</p>
10	RW	0x0	<p>grf_lvds_con_den_polarity</p> <p>LVDS controller den_polarity signal control</p>
9	RW	0x0	<p>grf_lvds_con_hs_polarity</p> <p>LVDS controller hs_polarity signal control</p>
8	RW	0x0	<p>grf_lvds_con_clkinv</p> <p>LVDS controller clkinv signal control</p>
7	RW	0x0	<p>grf_lvds_con_startphase</p> <p>LVDS controller startphase signal control</p>
6	RW	0x0	<p>grf_lvds_con_ttl_en</p> <p>LVDS controller ttl_en signal control</p>
5	RW	0x0	<p>grf_lvds_con_startsel</p> <p>LVDS controller startsel signal control</p>
4	RW	0x0	<p>grf_lvds_con_chasel</p> <p>LVDS controller chasel signal control</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	grf_lvds_con_msbsel LVDS controller msbsel signal control
2:0	RW	0x0	grf_lvds_con_select LVDS controller select signal control

**GRF\_SOC\_CON8**

Address: Operational Base + offset (0x0264)

SoC control register 8

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RW	0x0	grf_edp_hdcp_protect eDP HDCP function protection 1'b1: protect 1'b0: not protect
14	RW	0x0	grf_edp_bist_en eDP PHY BIST function enabled 1'b1: enable 1'b0: disable
13	RW	0x0	grf_edp_mem_ctrl_sel eDP memory control selection 1'b1: controlled by eDP controller internal logic 1'b0: controlled by APB BUS
12	RW	0x0	grf_hdmi_cec_mux_sel HDMI cec source selection 1'b1: from GPIO7C[0] 1'b0: from GPIO7C[7]

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:8	RW	0x0	grf_dphy_tx0_forcetxstopmode MIPI DPHY TX0 force lane into transmit mode and generate stop state. Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.
7:4	RW	0x0	grf_dphy_tx0_forcerxmode MIPI DPHY TX0 force lane into receive mode/wait for stop stat. Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.
3:0	RW	0xe	grf_dphy_tx0_turndisable MIPI DPHY TX0 disable turn around control Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.

**GRF\_SOC\_CON9**

Address: Operational Base + offset (0x0268)

SoC control register 9

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:12	RW	0x0	grf_dphy_tx1rx1_enable MIPI DPHY TX1RX1 enable lane N module(N=0~3). Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.
11:8	RW	0x0	grf_dphy_tx1rx1_forcetxstopmode MIPI DPHY TX1RX1 force lane into transmit mode and generate stop state. Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:4	RW	0x0	grf_dphy_tx1rx1_forcerxmode MIPI DPHY TX1RX1 force lane into receive mode/wait for stop stat. Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.
3:0	RW	0xe	grf_dphy_tx1rx1_turndisable MIPI DPHY TX1RX1 disable turn around control Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.

**GRF\_SOC\_CON10**

Address: Operational Base + offset (0x026c)

SoC control register 10

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:12	RW	0x0	grf_dphy_rx0_enable MIPI DPHY RX0 enable lane N module(N=0~3). Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.
11:8	RW	0x0	grf_dphy_rx0_forcetxstopmode MIPI DPHY RX0 force lane into transmit mode and generate stop sate. Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.
7:4	RW	0x0	grf_dphy_rx0_forcerxmode MIPI DPHY RX0 force lane into receive mode/wait for stop stat. Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0xf	grf_dphy_rx0_turndisable MIPI DPHY RX0 disable turn around control Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.

**GRF\_SOC\_CON11**

Address: Operational Base + offset (0x0270)

SoC control register 11

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RW	0x0	gpio8_a2_fall_edge_irq_pd GII08A[2] fall edge interrupt pending status 1'b1: enable 1'b0: disable
14	RW	0x0	gpio8_a2_fall_edge_irq_en GII08A[2] fall edge interrupt enable 1'b1: enable 1'b0: disable
13	RW	0x0	gpio8_a2_rise_edge_irq_pd GII08A[2] rise edge interrupt pending status 1'b1: enable 1'b0: disable
12	RW	0x0	gpio8_a2_rise_edge_irq_en GII08A[2] rise edge interrupt enable 1'b1: enable 1'b0: disable
11	RW	0x0	gpio7_c6_fall_edge_irq_pd GII07C[6] fall edge interrupt pending status 1'b1: enable 1'b0: disable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
10	RW	0x0	gpio7_c6_fall_edge_irq_en GIO7C[6] fall edge interrupt enable 1'b1: enable 1'b0: disable
9	RW	0x0	gpio7_c6_rise_edge_irq_pd GIO7C[6] rise edge interrupt pending status 1'b1: enable 1'b0: disable
8	RW	0x0	gpio7_c6_rise_edge_irq_en GIO7C[6] rise edge interrupt enable 1'b1: enable 1'b0: disable
7	RW	0x0	gpio7_b3_fall_edge_irq_pd GIO7B[3] fall edge interrupt pending status 1'b1: enable 1'b0: disable
6	RW	0x0	gpio7_b3_fall_edge_irq_en GIO7B[3] fall edge interrupt enable 1'b1: enable 1'b0: disable
5	RW	0x0	gpio7_b3_rise_edge_irq_pd GIO7B[3] rise edge interrupt pending status 1'b1: enable 1'b0: disable
4	RW	0x0	gpio7_b3_rise_edge_irq_en GIO7B[3] rise edge interrupt enable 1'b1: enable 1'b0: disable
3	RW	0x0	sd_detectn_fall_edge_irq_pd sdmmc detect_n fall edge interrupt pending status 1'b1: enable 1'b0: disable
2	RW	0x0	sd_detectn_fall_edge_irq_en sdmmc0 detect_n signal fall edge interrupt enable 1'b1: enable 1'b0: disable
1	RW	0x0	sd_detectn_rise_edge_irq_pd sdmmc detect_n rise edge interrupt pending status 1'b1: enable 1'b0: disable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	sd_detectn_rise_edge_irq_en sdmmc0 detect_n signal rise edge interrupt enable 1'b1: enable 1'b0: disable

**GRF\_SOC\_CON12**

Address: Operational Base + offset (0x0274)

SoC control register 12

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:7	RW	0x000	grf_edp_frq_vid_ck_in eDP PHY frequency information of vid_ck_in frq_vid_ck_in<8:0>/8 = freq(vid_ck_in)/10
6	RW	0x0	grf_edp_vid_lock eDP PHY input video PLL stable indicator 1'b1: stable 1'b0: unstable
5	RW	0x0	grf_edp_iddq_en eDP PHY IDDQ enable 1'b0: disable 1'b1: enable, all circuits are power down, all IO are high-z
4	RW	0x1	grf_edp_ref_clk_sel eDP PHY reference clock source selection 1'b0: from PAD(IO_EDP_OSC_CLK_24M) 1'b1: from internal 24MHz or 27MHz clock
3	RW	0x0	grf_edp_dc_tp_i eDP PHY analog DC test point input
2	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RW	0x3	grf_filter_cnt_sel the counter select for sd card detect filter 2'b00: 5ms 2'b01: 15ms 2'b10: 35ms 2'b11: 50ms

**GRF\_SOC\_CON13**

Address: Operational Base + offset (0x0278)

SoC control register 13

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:12	RW	0x0	grf_edp_tx_bscan_data eDP TX boundary data bit0: boundary data to ch0 bit1: boundary data to ch1 bit2: boundary data to ch2 bit3: boundary data to ch3
11	RW	0x0	grf_edp_tx_bscan_en eDP TX boundary enable 1'b0: disable 1'b1: enable
10	RO	0x0	reserved
9:5	RW	0x00	grf_uart_rts_sel UART polarity selection for rts port Every bit for one UART, bit4 is for UART_EXP, bit3 is for UART_GPS, bit2 is for UART_DBG, bit1 is for UART_BB, bit0 is for UART_BT. 1'b1: high asserted 1'b0: low asserted

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:0	RW	0x00	grf_uart_cts_sel UART polarity selection for cts port Every bit for one UART, bit4 is for UART_EXP, bit3 is for UART_GPS, bit2 is for UART_DBG, bit1 is for UART_BB, bit0 is for UART_BT. 1'b1: high asserted 1'b0: low asserted

**GRF\_SOC\_CON14**

Address: Operational Base + offset (0x027c)

SoC control register 14

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RW	0x0	grf_dphy_tx1rx1_basedir MIPI DPHY TX1RX1 base direction control
14	RW	0x0	grf_dphy_tx1rx1_masterslavez MIPI DPHY TX1RX1 master/slave control
13	RW	0x0	dphy_rx1_src_sel MIPI DPHY RX1 source selection 1'b1: isp 1'b0: csi host
12	RW	0x0	dphy_tx1rx1_enabledclk MIPI DPHY TX1RX1 enable clock Lane module
11	RO	0x0	reserved
10:3	RW	0x00	dphy_rx0_testdin MIPI DPHY RX0 test bus input data
2	RW	0x0	dphy_rx0_testen MIPI DPHY RX0 test bus enable
1	RW	0x0	dphy_rx0_testclk MIPI DPHY RX0 test bus clock

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	dphy_rx0_testclr MIPI DPHY RX0 test bus clear control

**GRF\_SOC\_STATUS0**

Address: Operational Base + offset (0x0280)

SoC status register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	ddrupctl1_bbflags DDR channel 1 NIF output vector which provides combined information about the status of each memory bank. The de-assertion is based on when precharge, activates, reads/writes. Bit0 indication Bank0 busy, bit1 indication Bank1 busy, and so on.
15:0	RW	0x0000	ddrupctl0_bbflags DDR channel 0 NIF output vector which provides combined information about the status of each memory bank. The de-assertion is based on when precharge, activates, reads/writes. Bit0 indication Bank0 busy, bit1 indication Bank1 busy, and so on.

**GRF\_SOC\_STATUS1**

Address: Operational Base + offset (0x0284)

SoC status register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	gmac_portselect MAC Port Select A high indicates an MII interface, and a low a GMII interface.
30:26	RO	0x0	reserved
25:22	RW	0x0	hsic_stat_ehci_lpsmc_state HSIC ehci_lpsmc_state bit status
21:16	RW	0x00	hsic_stat_ehci_usbsts HSIC ehci_usbsts bit status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:13	RW	0x0	ddrupctl1_stat 3'b000: Init_mem 3'b001: Config 3'b010: Config_req 3'b011: Access 3'b100: Access_req 3'b101: Low_power 3'b110: Low_power_entry_req 3'b111: Low_power_exit_req
12:10	RW	0x0	ddrupctl0_stat 3'b000: Init_mem 3'b001: Config 3'b010: Config_req 3'b011: Access 3'b100: Access_req 3'b101: Low_power 3'b110: Low_power_entry_req 3'b111: Low_power_exit_req
9	RW	0x0	newpll_lock NEW PLL lock status
8	RW	0x0	generalpll_lock GENERAL PLL lock status
7	RW	0x0	codecpll_lock CODEC PLL lock status
6	RW	0x0	armpll_lock ARM PLL lock status
5	RW	0x0	ddrpll_lock DDR PLL lock status
4	RW	0x0	newpll_clk NEW PLL clock output
3	RW	0x0	generalpll_clk GENERAL PLL clock output
2	RW	0x0	codecpll_clk CODEC PLL clock output
1	RW	0x0	armpll_clk ARM PLL clock output
0	RW	0x0	ddrpll_clk DDR PLL clock output

**GRF\_SOC\_STATUS2**

Address: Operational Base + offset (0x0288)

SoC status register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
30	RW	0x0	usbhost0_stat_ohci_bufacc USB HOST0 ohci_bufacc signal status
29	RW	0x0	usbhost0_stat_ohci_rmtwkp USB HOST0 ohci_rmtwkp signal status
28:27	RW	0x0	usbhost0_utmi_linestate USB HOST0 utmi_linestate signal status
26	RW	0x0	usbhost0_stat_ohci_drwe USB HOST0 ohci_drwe signal status
25	RW	0x0	usbhost0_stat_ohci_rwe USB HOST0 ohci_rwe signal status
24	RW	0x0	usbhost0_stat_ohci_ccs USB HOST0 ohci_ccs signal status
23	RW	0x0	usbhost1_utmiotg_iddig USB HOST1 utmiotg_iddig signal status
22:21	RW	0x0	usbhost1_utmi_linestate USB HOST1 utmi_linestate signal status
20	RW	0x0	usbhost1_utmisrp_bvalid USB HOST1 utmisrp_bvalid signal status
19	RW	0x0	usbhost1_utmiotg_vbusvalid USB HOST1 utmiotg_vbusvalid signal status
18	RW	0x0	usbhost1_chirp_on USB HOST1 chirp_on signal status
17	RW	0x0	usbotg_utmiotg_iddig USB OTG utmiotg_iddig signal status
16:15	RW	0x0	usbotg_utmi_linestate USB OTG utmi_linestate signal status
14	RW	0x0	usbotg_utmisrp_bvalid USB OTG utmisrp_bvalid
13	RW	0x0	usbotg_utmiotg_vbusvalid USB OTG utmiotg_vbusvalid signal status
12	RO	0x0	reserved
11	RW	0x0	hsic_stat_ehci_xfer_prdc HSIC ehci_xfer_prdc signal status
10:0	RW	0x000	hsic_stat_ehci_xfer_cnt HSIC ehci_xfer_cnt signal status

**GRF\_SOC\_STATUS3**

Address: Operational Base + offset (0x028c)

SoC status register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	nif0_fifo0 DDR channel0 NIF interface FIFO0 status

**GRF\_SOC\_STATUS4**

Address: Operational Base + offset (0x0290)

SoC status register 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	nif0_fifo1 DDR channel0 NIF interface FIFO1 status

### **GRF\_SOC\_STATUS5**

Address: Operational Base + offset (0x0294)

SoC status register 5

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	nif0_fifo2 DDR channel0 NIF interface FIFO2 status

### **GRF\_SOC\_STATUS6**

Address: Operational Base + offset (0x0298)

SoC status register 6

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	nif0_fifo3 DDR channel0 NIF interface FIFO3 status

### **GRF\_SOC\_STATUS7**

Address: Operational Base + offset (0x029c)

SoC status register 7

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	nif1_fifo0 DDR channel1 NIF interface FIFO0 status

### **GRF\_SOC\_STATUS8**

Address: Operational Base + offset (0x02a0)

SoC status register 8

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	nif1_fifo1 DDR channel1 NIF interface FIFO1 status

### **GRF\_SOC\_STATUS9**

Address: Operational Base + offset (0x02a4)

SoC status register 9

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	nif1_fifo2 DDR channel1 NIF interface FIFO2 status

### **GRF\_SOC\_STATUS10**

Address: Operational Base + offset (0x02a8)

SoC status register 10

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	nif1_fifo3 DDR channel1 NIF interface FIFO3 status

**GRF\_SOC\_STATUS11**

Address: Operational Base + offset (0x02ac)

SoC status register 11

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi0_eff_wr_num DDR channel0 DFI interface write command number

**GRF\_SOC\_STATUS12**

Address: Operational Base + offset (0x02b0)

SoC status register 12

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi0_eff_rd_num DDR channel0 DFI interface read command number

**GRF\_SOC\_STATUS13**

Address: Operational Base + offset (0x02b4)

SoC status register 13

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi0_eff_act_num DDR channel0 DFI interface active command number

**GRF\_SOC\_STATUS14**

Address: Operational Base + offset (0x02b8)

SoC status register 14

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi0_timer_val DDR channel0 DFI interface statistics timer value

**GRF\_SOC\_STATUS15**

Address: Operational Base + offset (0x02bc)

SoC status register 15

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi1_eff_wr_num DDR channel1 DFI interface write command number

**GRF\_SOC\_STATUS16**

Address: Operational Base + offset (0x02c0)

SoC status register 16

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi1_eff_rd_num DDR channel1 DFI interface read command number

### **GRF\_SOC\_STATUS17**

Address: Operational Base + offset (0x02c4)

SoC status register 17

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi1_eff_act_num DDR channel1 DFI interface active command number

### **GRF\_SOC\_STATUS18**

Address: Operational Base + offset (0x02c8)

SoC status register 18

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi1_timer_val DDR channel1 DFI interface statistics timer value

### **GRF\_SOC\_STATUS19**

Address: Operational Base + offset (0x02cc)

SoC status register 19

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	usbhost1_fsvminus USB HOST1 PHY fsvminus bit status
30	RW	0x0	usbhost1_fsvplus USB HOST1 PHY fsvplus bit status
29	RW	0x0	usbhost1_chgdet USB HOST1 PHY charge detect status
28	RW	0x0	usbhost0_fsvminus USB HOST0 PHY fsvminus bit status
27	RW	0x0	usbhost0_fsvplus USB HOST0 PHY fsvplus bit status
26	RW	0x0	usbhost0_chgdet USB HOST0 PHY charge detect status
25	RW	0x0	usbotg_fsvminus USB OTG PHY fsvminus bit status
24	RW	0x0	usbotg_fsvplus USB OTG PHY fsvplus bit status
23	RW	0x0	usbotg_chgdet USB OTG PHY charge detect status
22:14	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13:11	RW	0x0	host_l3_ocp_sconnect Host interface L3 OCP sconnect status
10	RW	0x0	host_l3_ocp_tactive Host interface L3 OCP tactive status
9:8	RW	0x0	host_l3_ocp_mconnect Host interface L3 OCP mconnect status
7	RW	0x0	host_wakeack Host interface wakeack status
6:5	RW	0x0	host_eoi_out Host interface eoi_out status
4	RW	0x0	host_mwait_out Host interface mwait_out status
3	RW	0x0	host_mwakeup Host interface mwakeup status
2	RW	0x0	host_mstandby Host interface mstandby status
1:0	RW	0x0	host_sidle_ack Host interface sidle ack

**GRF\_SOC\_STATUS20**

Address: Operational Base + offset (0x02d0)

SoC status register 20

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	host_gen0 Host interface geno bit stauts The GENI GEN0 is a mechanism that allows the 2 chip to exchange flags (interrupts), up to 32 independent flags are available.

**GRF\_SOC\_STATUS21**

Address: Operational Base + offset (0x02d4)

SoC status register 21

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RW	0x00	usbhost0_stat_ehci_usbsts USB host0 ehci_usbsts bit status
25:15	RW	0x000	usbhost0_stat_ehci_xfer_cnt USB host0 ehci_xfer counter status
14	RW	0x0	usbhost0_stat_ehci_xfer_prdc USB host0 ehci_xfer_prdc bit status
13:10	RW	0x0	usbhost0_stat_ehci_lpsmc_state USB host0 ehci_lpsmc_state bit status
9	RW	0x0	usbhost0_stat_ehci_bufacc USB host0 ehci_bufacc bit status
8	RW	0x0	usbhost0_stat_ohci_globalsuspend USB host0 ohci_globalsuspend bit status

Bit	Attr	Reset Value	Description
7:0	RW	0x00	dphy_rx0_testdout MIPI DPHY RX0 test bus data output

**GRF\_PERIDMAC\_CON0**

Address: Operational Base + offset (0x02e0)

PERI DMAC control register 0

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	wirte_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:8	RW	0x00	peridmac_boot_addr peridmac_boot_addr[19:12] PERI DMAC boot_addr[19:12] input control Configures the address location that contains the first instruction the DMAC executes, when it exits from reset.
7:4	RW	0xf	peridmac_boot_periph_ns peridmac_boot_periph_ns[19:16] PERI DMAC boot_peri_ns input control Controls the security state of a peripheral request interface, when the PERI DMAC exits from reset. Note: PERI DMAC don't support secure feature, these bits don't need to be configured
3	RW	0x1	peridmac_boot_manager_ns PERI DMAC boot_manager_ns input control When the DMAC exits from reset , this signal controls the security state of the DMA manager thread: 1'b0: assigns DMA manager to the secure state 1'b1: assigns DMA manager to the Non-secure state

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:1	RW	0x1	grf_drtype_peridmac PERI DMAC type of acknowledgement or request for peripheral signals: 2'b00: single level request 2'b01: burst level request 2'b10: acknowledging a flush request 2'b11: reserved
0	RW	0x0	peridmac_boot_from_pc PERI DMAC boot_from_pc input control Controls the location in which the DMAC0 executes its initial instruction, after it exits from reset : 1'b0: DMAC waits for an instruction from APB interface 1'b1: DMAC manager thread executes the instruction that is located at the address that boot_addr[31:0] provided.

**GRF\_PERIDMAC\_CON1**

Address: Operational Base + offset (0x02e4)

PERI DMAC control register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	wirte_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:12	RO	0x0	reserved
11:0	RW	0x000	peridmac_boot_addr peridmac_boot_addr[31:20] PERI DMAC boot_addr[31:20] input control Configures the address location that contains the first instruction the DMAC executes, when it exits from reset.

**GRF\_PERIDMAC\_CON2**

Address: Operational Base + offset (0x02e8)

PERI DMAC control register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	wirte_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0xffff	peridmac_boot_irq_ns PERI DMAC boot_irq_ns input control Controls the security state of an event-interrupt resource , when the PERI DMAC exits from reset. Note : PERI DMAC don't support secure feature, these bits don't need to be configured.

**GRF\_PERIDMAC\_CON3**

Address: Operational Base + offset (0x02ec)

PERI DMAC control register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	WO	0x0000	wirte_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0xffff	<p>peridmac_boot_periph_ns  PERI DMAC boot_peri_ns input control  Controls the security state of a peripheral request interface, when the DMAC exits from reset.</p> <p>Note: PERI DMAC don't support secure feature, these bits don't need to be configured.</p>

**GRF\_DDRC0\_CON0**

Address: Operational Base + offset (0x02f0)

DDRC0 control register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable  bit0~15 write enable  When bit 16=1, bit 0 can be written by software .  When bit 16=0, bit 0 cannot be written by software;  When bit 17=1, bit 1 can be written by software .  When bit 17=0, bit 1 cannot be written by software;  .....  When bit 31=1, bit 15 can be written by software .  When bit 31=0, bit 15 cannot be written by software;</p>
15:13	RO	0x0	reserved
12:11	RW	0x0	ddr0(dto_lb) DDR0 DTO I/O internal loopback enable
10:9	RW	0x0	ddr0(dto_te) DDR0 DTO I/O on-die termination enable
8:7	RW	0x0	ddr0(dto_pdr) DDR0 DTO I/O receiver power down
6:5	RW	0x0	ddr0(dto_pdd) DDR0 DTO I/O driver power down
4:3	RW	0x0	ddr0(dto_iom) DDR0 DTO I/O mode select
2:1	RW	0x0	ddr0(dto_oe) DDR0 DTO I/O output enable
0	RW	0x0	ddr0(ato_ae) Enables, if set, the analog test output I/O. Connects to the AE pin of the analog test output I/O

**GRF\_DDRC1\_CON0**

Address: Operational Base + offset (0x02f4)

DDRC1 control register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:13	RO	0x0	reserved
12:11	RW	0x0	ddr1(dto_lb) DDR1 DTO I/O internal loopback enable
10:9	RW	0x0	ddr1(dto_te) DDR1 DTO I/O on-die termination enable
8:7	RW	0x0	ddr1(dto_pdr) DDR1 DTO I/O receiver power down
6:5	RW	0x0	ddr1(dto_pdd) DDR1 DTO I/O driver power down
4:3	RW	0x0	ddr1(dto_iom) DDR1 DTO I/O mode select
2:1	RW	0x0	ddr1(dto_oe) DDR1 DTO I/O output enable
0	RW	0x0	ddr1(ato_ae) Enables, if set, the analog test output I/O. Connects to the AE pin of the analog test output I/O

**GRF\_CPU\_CON0**

Address: Operational Base + offset (0x02f8)

CPU control register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15	RW	0x1	cfgaddrfilt_en_grf A17 cfgaddrfilt_en bit control
14	RO	0x0	reserved
13:10	RW	0x0	<p>cfgend_a17</p> <p>A17 cfgend bit control</p> <p>Every bit for one core, bit3 is for core3, bit2 is for core2, bit1 is for core1, bit0 is for core0.</p>
9	RW	0x1	tpiu_ctl_grf tpiu_ctl bit control
8:5	RW	0x1	cs_instid_grf Coresight cs_instid bit control
4	RW	0x0	I2rstdisable_grf A17 I2rstdisable bit control
3:0	RW	0x0	I1rstdisable_grf A17 I1rstdisable bit control
			Every bit for one core, bit3 is for core3, bit2 is for core2, bit1 is for core1, bit0 is for core0.

**GRF\_CPU\_CON1**

Address: Operational Base + offset (0x02fc)

CPU control register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x0ff0	cfgaddrfilt_start_grf A17 non secure filter start address[15:0]

**GRF\_CPU\_CON2**

Address: Operational Base + offset (0x0300)

CPU control register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x0fff	cfgaddrfilt_end_grf A17 non secure filter end address[15:0]

**GRF\_CPU\_CON3**

Address: Operational Base + offset (0x0304)

CPU control register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:12	RO	0x0	reserved
11:8	RW	0x0	<p>cfgnmfi_a17</p> <p>A17 cfgnmfi bit control</p> <p>Every bit for one core, bit3 is for core3, bit2 is for core2, bit1 is for core1, bit0 is for core0.</p>
7:4	RW	0x0	<p>cfgaddrfilt_end_grf</p> <p>A17 non secure filter end address[19:16]</p>
3:0	RW	0x0	<p>cfgaddrfilt_start_grf</p> <p>A17 non secure filter start address[19:16]</p>

**GRF\_CPU\_CON4**

Address: Operational Base + offset (0x0308)

CPU control register 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:13	RW	0x1	l2_mem_ema_grf L2 memory EMA control
12:10	RW	0x1	owl_mem_ema_grf A17 memory EMA control
9	RW	0x0	evento_clear A17 evento clear bit control
8	RW	0x0	eventi_a17 A17 eventi bit control
7:4	RO	0x0	reserved
3:0	RW	0x0	teinit_a17 A17 teinit bit control Every bit for one core, bit3 is for core3, bit2 is for core2, bit1 is for core1, bit0 is for core0.

**GRF\_CPU\_STATUS0**

Address: Operational Base + offset (0x0318)

CPU status register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	RO	0x0	reserved
14	RW	0x0	evento_rising_edge evento signal rising edge
13:10	RW	0x0	owl_pmupl1_grf A17 PMU Privilege level 1 event Every bit for one core, bit3 is for core3, bit2 is for core2, bit1 is for core1, bit0 is for core0.
9:6	RW	0x0	owl_pmupl2_grf A17 PMU Privilege level 2 event Every bit for one core, bit3 is for core3, bit2 is for core2, bit1 is for core1, bit0 is for core0.
5:2	RW	0x0	owl_pmusecure_grf A17 pmu secure event Every bit for one core, bit3 is for core3, bit2 is for core2, bit1 is for core1, bit0 is for core0.
1	RW	0x0	jtagnsw_st_grf JTAG nsw status
0	RW	0x0	jtagtop_st_grf JTAG top status

**GRF\_UOC0\_CON0**

Address: Operational Base + offset (0x0320)

UOC0 control register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15	RW	0x0	usb0tg_linestate_irq_pd USB OTG linestate interrupt pending bit
14	RW	0x0	usb0tg_linestate_irq_en USB OTG line state interrupt enable
13	RW	0x0	<p>usb0tg_siddq</p> <p>USB OTG IDDQ test enable</p> <p>This test signal enables you to perform IDDQ testing by powering down all analog blocks.</p> <p>1'b1: The analog blocks are powered down.</p> <p>1'b0: The analog blocks are powered up.</p>
12	RW	0x0	<p>usb0tg_port_reset</p> <p>USB OTG per-port reset</p> <p>When asserted, this customer-specific signal resets the corresponding port transmit and receive logic without disabling the clocks within the PHY.</p> <p>1'b1: The transmit and receive finite state machines (FSMs) are reset, and the line_state logic combinatorially reflects the state of the single-ended receivers.</p> <p>1'b0: The transmit and receive FSMs are operational, and the line_state logic becomes sequential after 11 PHYCLOCK cycles.</p>
11:10	RO	0x0	reserved
9:8	RW	0x0	usb0tg_scaledown_mode USB OTG scale down mode control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:5	RW	0x4	<b>usbotg_tune</b> USB OTG VBUS valid threshold adjustment This bus adjusts the voltage level for the VBUS Valid threshold. 3'b111: +9% 3'b110: +6% 3'b101: +3% 3'b100: Design default 3'b011: -3% 3'b010: -6% 3'b001: -9% 3'b000: -12%
4	RW	0x0	<b>usbotg_disable</b> USB OTG block disable 1'b1: the USB OTG block is power down 1'b0: the USB OTG block is power up
3:1	RW	0x4	<b>usbotg_compdistune</b> Disconnect Threshold Adjustment This bus adjusts the voltage level for the threshold used to detect a disconnect event at the host. 3'b111: +4.5% 3'b110: +3% 3'b101: +1.5% 3'b100: Design default 3'b011: -1.5% 3'b010: -3% 3'b001: -4.5% 3'b000: -6%
0	RW	0x1	<b>usbotg_common_on_n</b> USB OTG common block power-down control This signal controls the power-down signals in the XO, Bias, and PLL blocks when the USB 2.0 PHY is in Suspend or Sleep mode. 1'b1: In Suspend mode, the XO, Bias, and PLL blocks are powered down. In Sleep mode, the Bias and PLL blocks are powered down. 1'b0: In Suspend mode, the XO, Bias, and PLL blocks remain powered in Suspend mode. In Sleep mode, if the reference clock is a crystal, the XO block remains powered.

**GRF\_UOC0\_CON1**

Address: Operational Base + offset (0x0324)  
UOC0 control register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:14	RW	0x1	<p>usbotg_txrisetune</p> <p>USB OTG HS transmitter rise/fall time adjustment</p> <p>This bus adjusts the rise/fall times of the high-speed waveform.</p> <p>2'b11: -20%</p> <p>2'b10: -15%</p> <p>2'b01: design default</p> <p>2'b00: +10%</p>
13:12	RW	0x3	<p>usbotg_txhsxvtune</p> <p>USB OTG transmitter high-speed crossover adjustment</p> <p>This bus adjusts the voltage at which the DP and DM signals cross while transmitting in HS mode.</p> <p>2'b11: Default setting</p> <p>2'b10: +15 mV</p> <p>2'b01: -15 mV</p> <p>2'b00: Reserved</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:8	RW	0x3	<p>usbotg_txvreftune            USB OTG HS DC voltage level adjustment            This bus adjusts the high-speed DC level voltage.</p> <p>4'b1111: +8.75%            4'b1110: +7.5%            4'b1101: +6.25%            4'b1100: +5%            4'b1011: +3.75%            4'b1010: +2.5%            4'b1001: +1.25%            4'b1000: Design default            4'b0111: -1.25%            4'b0110: -2.5%            4'b0101: -3.75%            4'b0100: -5%            4'b0011: -6.25%            4'b0010: -7.5%            4'b0001: -8.75%            4'b0000: -10%</p>
7:4	RW	0x3	<p>usbotg_txfslstune            USB OTG FS/LS source impedance adjustment            This bus adjusts the low- and full-speed single-ended source impedance while driving high. The following adjustment values are based on nominal process, voltage, and temperature.</p> <p>4'b1111: -5%            4'b0111: -2.5%            4'b0011: Design default            4'b0001: +2.5%            4'b0000: +5%</p>
3	RW	0x0	<p>usbotg_txpreempppulse<sup>tune</sup>            USB OTG HS transmitter pre-emphasis duration control            This signal controls the duration for which the HS pre-emphasis current is sourced onto DP0 or DM0. transition in HS mode.</p> <p>1'b1: 1X, short pre-emphasis current duration            1'b0: (desian default) 2X, long pre-emphasis currrent duration</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:0	RW	0x3	usbotg_sqrxtune USB OTG squelch threshold adjustment This bus adjusts the voltage level for the threshold used to detect valid high-speed data. 3'b111: -20% 3'b110: -15% 3'b101: -10% 3'b100: -5% 3'b011: Design default 3'b010: +5% 3'b001: +10% 3'b000: +15%

**GRF\_UOC0\_CON2**

Address: Operational Base + offset (0x0328)

UOC0 control register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RW	0x0	usbotg_acaenb USB OTG ACA ID_OTG pin resistance detection enable 1'b1: enable detection on resistance on the ID_OTG pin of an ACA 1'b0: disable detection on resistance on the ID_OTG pin of an ACA

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14	RW	0x0	usbotg_dcdenb USB OTG data contact detection enable 1'b1: IDP_SRC current is sourced onto DP, pull-down resistance on DMA is enabled 1'b0: IDP_SRC current is disable, pull-down resistance on DM is disabled
13	RO	0x0	reserved
12:11	RW	0x1	usbotg_txrestune USB OTG source impedance adjustment 2'b11: source impedance is decreased by 4ohm 2'b10: source impedance is decreased by 2ohm 2'b01: design default 2'b00: source impedance is decreased by 1.5ohm
10	RW	0x1	usbotg_sleepm USB OTG sleep mode enable Asserting this signal place the USB PHY in sleep mode. 1'b0: sleep mode enable 1'b1: normal mode
9	RO	0x0	reserved
8	RW	0x1	usbotg_retenable_n USB OTG retention mode enable 0: retention mode enable 1: retention mode disable
7	RW	0x0	usbotg_vdatsrcenb USB OTG battery charging sourcing select 1'b1: data source voltage is enable 1'b0: data source voltage is disable
6	RW	0x0	usbotg_vdatdetenb USB OTG battery charging attach/connect detection enable 1'b1: enable 1'b0: disable
5	RW	0x0	usbotg_chrgsel USB OTG battery charging source select 1'b1: data source voltage is sourced onto DM and sunk from DP 1'b0: data source voltage is sourced onto DP and sunk from DM

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:3	RW	0x1	usbotg_txpreempampntune 2'b11: 3X pre-emphasis current 2'b10: 2X pre-emphasis current 2'b01: 1X pre-emphasis current 2'b00: HS Transmitter Pre-Emphasis is disabled
2	RW	0x0	usbotg_soft_con_sel 1'b0: software control usb otg disable 1'b1: software control usb otg enable
1	RW	0x0	usbotg_vbusvldextsel USB OTG external VBUS valid select This signal selects the VBUSVLDEXT input or the internal Session Valid comparator to indicate when the VBUS signal on the USB cable is valid. 1'b1: The VBUSVLDEXT input is used. 1'b0: The internal Session Valid comparator is used.
0	RW	0x0	usbotg_vbusvldext USB OTG external VBUS valid indicator This signal is valid in Device mode and only when the VBUSVLDEXTSEL signal is set to 1. VBUSVLDEXT indicates whether the VBUS signal on the USB cable is valid. In addition, BUSVLDEXT enables the pullup resistor on the D+ line. 1'b1: The VBUS signal is valid, and the pull-up resistor on D+ is enabled. 1'b0: The VBUS signal is not valid, and the pull-up resistor on D+ is disabled.

**GRF\_UOC0\_CON3**

Address: Operational Base + offset (0x032c)

UOC0 control register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15	RW	0x0	usb0tg_dbnce_filt_bypass USB OTG debounce filter bypass enable
14	RO	0x0	reserved
13	RW	0x0	usb0tg_iddig_sel USB OTG iddig soft control enable 1'b1: software control 1'b0: hardware control
12	RW	0x0	usb0tg_iddig USB OTG iddig software control bit
11:8	RO	0x0	reserved
7	RW	0x0	usb0tg_bypasssel transmitter digital bypass select 1'b1: transmitter digital bypass mode is enabled 1'b0: transmitter digital bypass mode is disabled
6	RW	0x0	usb0tg_bypassdmen DM0 transmitter digital bypass enable 1'b1: DM0 FS/LS driver is enabled and driven with the BYPASSDPDATA0 signals 1'b0: DM0 FS/LS driver is disabled in transmitter digital bypass mode
5	RW	0x0	usb0tg_utmi_termselect USB OTG utmi termination select 1'b1: full speed terminations are enabled 1'b0: high speed terminations are enabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:3	RW	0x0	usbotg_utmi_xcvrselect USB OTG utmi transceiver select 2'b11: sends an LS packet on an FS bus or receives an LS packet 2'b10: LS transceiver 2'b01: FS transceiver 2'b00: HS transceiver
2:1	RW	0x0	usbotg_utmi_opmode USB OTG utmi operation mode This controller bus selects the UTMI+ operation mode 2'b11: normal operation without SYNC or EOP generation 2'b10: disable bit stuffing and NRZI encoding 2'b01: no-driving 2'b00: normal
0	RW	0x1	usbotg_utmi_suspend_n USB OTG suspend mode enable 1'b1: normal operation mode 1'b0: suspend mode

**GRF\_UOC0\_CON4**

Address: Operational Base + offset (0x0330)

UOC0 control register 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:8	RO	0x0	reserved
7	RW	0x0	usbotg_id_fall_edge_irq_pd USB OTG id fall edge interrupt pending bit, write 1 to this bit , it will be cleared.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RW	0x0	usb0tg_id_fall_edge_irq_en USB OTG id fall edge interrupt enable
5	RW	0x0	usb0tg_id_rise_edge_irq_pd USB OTG id rise edge interrupt pending bit, write 1 to this bit , it will be cleared.
4	RW	0x0	usb0tg_id_rise_edge_irq_en USB OTG id rise edge interrupt enable
3	RW	0x0	usb0tg_bvalid_irq_pd USB OTG bvalid interrupt pending bit, write 1 to this bit, it will be cleared.
2	RW	0x0	usb0tg_bvalid_irq_en USB OTG bvalid interrupt enable
1:0	RW	0x3	linestate_cnt_sel linestate signal filter time select 2'b00: 100us 2'b01: 500us 2'b10: 2.5ms 2'b11: 15ms

**GRF\_UOC1\_CON0**

Address: Operational Base + offset (0x0334)

UOC1 control register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RW	0x0	usbhost0_linestate_irq_pd USB HOST0 linestate interrupt pending bit
14	RW	0x0	usbhost0_linestate_irq_en USB HOST0 line state interrupt enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13	RW	0x0	usbhost0_siddq USB HOST0 IDDQ test enable This test signal enables you to perform IDDQ testing by powering down all analog blocks. 1'b1: The analog blocks are powered down. 1'b0: The analog blocks are powered up.
12	RW	0x0	usbhost0_port_reset USB HOST0 per-port reset When asserted, this customer-specific signal resets the corresponding port transmit and receive logic without disabling the clocks within the PHY. 1'b1: The transmit and receive finite state machines (FSMs) are reset, and the line_state logic combinatorially reflects the state of the single-ended receivers. 1'b0: The transmit and receive FSMs are operational, and the line_state logic becomes sequential after 11 PHYCLOCK cycles.
11	RW	0x1	usbhost0_word_if USB HOST0 word_if bit control
10	RW	0x0	usbhost0_sim_mode USB HOST0 sim_mode bit control
9	RW	0x1	usbhost0_incrx_en USB HOST0 incr_x_en bit control
8	RW	0x1	usbhost0_incr8_en USB HOST0 incr8_en bit control
7:5	RW	0x4	usbhost0_tune USB HOST0 VBUS valid threshold adjustment This bus adjusts the voltage level for the VBUS Valid threshold. 3'b111: +9% 3'b110: +6% 3'b101: +3% 3'b100: Design default 3'b011: -3% 3'b010: -6% 3'b001: -9% 3'b000: -12%
4	RW	0x0	usbhost0_disable USB HOST0 block disable 1'b1: the USB HOST0 block is power down 1'b0: the USB HOST0 block is power up

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:1	RW	0x4	<p>usbhost0_compdistune            USB HOST0 disconnect threshold adjustment            This bus adjusts the voltage level for the threshold used to detect a disconnect event at the host.</p> <p>3'b111: +4.5%            3'b110: +3%            3'b101: +1.5%            3'b100: Design default            3'b011: -1.5%            3'b010: -3%            3'b001: -4.5%            3'b000: -6%</p>
0	RW	0x1	<p>usbhost0_common_on_n            USB HOST0 common block power-down control</p> <p>This signal controls the power-down signals in the XO, Bias, and PLL blocks when the USB 2.0 PHY is in Suspend or Sleep mode.</p> <p>1'b1: In Suspend mode, the XO, Bias, and PLL blocks are powered down. In Sleep mode, the Bias and PLL blocks are powered down.</p> <p>1'b0: In Suspend mode, the XO, Bias, and PLL blocks remain powered in Suspend mode. In Sleep mode, if the reference clock is a crystal, the XO block remains powered.</p>

**GRF\_UOC1\_CON1**

Address: Operational Base + offset (0x0338)

UOC1 control register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:14	RW	0x1	<p>usbhost0_txrisetune</p> <p>USB HOST0 HS transmitter rise/fall time adjustment</p> <p>This bus adjusts the rise/fall times of the high-speed waveform.</p> <p>2'b11: -20%</p> <p>2'b10: -15%</p> <p>2'b01: design default</p> <p>2'b00: +10%</p>
13:12	RW	0x3	<p>usbhost0_txhsxvtune</p> <p>USB HOST0 transmitter high-speed crossover adjustment</p> <p>This bus adjusts the voltage at which the DP and DM signals cross while transmitting in HS mode.</p> <p>2'b11: Default setting</p> <p>2'b10: +15 mV</p> <p>2'b01: -15 mV</p> <p>2'b00: Reserved</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:8	RW	0x3	<p>usbhost0_txvreftune            USB HOST0 HS DC voltage level adjustment            This bus adjusts the high-speed DC level voltage.</p> <p>4'b1111: +8.75%            4'b1110: +7.5%            4'b1101: +6.25%            4'b1100: +5%            4'b1011: +3.75%            4'b1010: +2.5%            4'b1001: +1.25%            4'b1000: Design default            4'b0111: -1.25%            4'b0110: -2.5%            4'b0101: -3.75%            4'b0100: -5%            4'b0011: -6.25%            4'b0010: -7.5%            4'b0001: -8.75%            4'b0000: -10%</p>
7:4	RW	0x3	<p>usbhost0_txfslstune            USB HOST0 FS/LS source impedance adjustment            This bus adjusts the low- and full-speed single-ended source impedance while driving high. The following adjustment values are based on nominal process, voltage, and temperature.</p> <p>4'b1111: -5%            4'b0111: -2.5%            4'b0011: Design default            4'b0001: +2.5%            4'b0000: +5%</p>
3	RW	0x0	<p>usbhost0_txpreemppulse<sup>tune</sup>            USB HOST0 HS transmitter pre-emphasis duration control            This signal controls the duration for which the HS pre-emphasis current is sourced onto DP0 or DM0. transition in HS mode.</p> <p>1'b1: 1X, short pre-emphasis current duration            1'b0: (desian default) 2X, long pre-emphasis currrent duration</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:0	RW	0x3	<p>usbhost0_sqrxtune  USB HOST0 squelch threshold adjustment  This bus adjusts the voltage level for the  threshold used to detect valid high-speed  data.</p> <p>3'b111: -20%  3'b110: -15%  3'b101: -10%  3'b100: -5%  3'b011: Design default  3'b010: +5%  3'b001: +10%  3'b000: +15%</p>

**GRF\_UOC1\_CON2**

Address: Operational Base + offset (0x033c)

UOC1 control register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable  bit0~15 write enable  When bit 16=1, bit 0 can be written by software .  When bit 16=0, bit 0 cannot be written by software;  When bit 17=1, bit 1 can be written by software .  When bit 17=0, bit 1 cannot be written by software;  .....  When bit 31=1, bit 15 can be written by software .  When bit 31=0, bit 15 cannot be written by software;</p>
15	RW	0x0	<p>usbhost0_acaenb  USB HOST0 ACA ID_OTG pin resistance detection enable  1'b1: enable detection on resistance on the ID_OTG pin of an ACA  1'b0: disable detection on resistance on the ID_OTG pin of an ACA</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14	RW	0x0	usbhost0_dcdenb USB HOST0 data contact detection enable 1'b1: IDP_SRC current is sourced onto DP, pull-down resistance on DMA is enabled 1'b0: IDP_SRC current is disable, pull-down resistance on DM is disabled
13	RW	0x0	usbhost0_app_prt_ovrcur USB HOST0 app_prt_ovrcur bit control
12:11	RW	0x1	usbhost0_txrestune USB HOST0 source impedance adjustment 2'b11: source impedance is desreased by 4ohm 2'b10: source impedance is desreased by 2ohm 2'b01: design default 2'b00: source impedance is desreased by 1.5ohm
10	RW	0x1	usbhost0_sleepm USB HOST0 sleep mode enable Asserting this signal place the USB PHY in sleep mode. 1'b0: sleep mode enable 1'b1: normal mode
9	RW	0x0	usbhost0_autoppd_on_overcur USB HOST0 autoppd_on_overcur bit control
8	RW	0x1	usbhost0_retenable_n USB HOST0 retention mode enable 0: retention mode enable 1: retention mode disable
7	RW	0x0	usbhost0_vdatsrcenb USB HOST0 battery charging sourcing select 1'b1: data source voltage is enable 1'b0: data source voltage is disable
6	RW	0x0	usbhost0_vdatdetenb USB HOST0 battery charging attach/connect detection enable 1'b1: enable 1'b0: disable
5	RW	0x0	usbhost0_chrgsel USB HOST0 battery charging source select 1'b1: data source voltage is sourced onto DM and sunk from DP 1'b0: data source voltage is sourced onto DP and sunk from DM

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:3	RW	0x1	usbhost0_txpreempampktune 2'b11: 3X pre-emphasis current 2'b10: 2X pre-emphasis current 2'b01: 1X pre-emphasis current 2'b00: HS Transmitter Pre-Emphasis is disabled
2	RW	0x0	usbhost0_soft_con_sel 1'b0: software control usb host0 disable 1'b1: software control usb host0 enable
1	RW	0x0	usbhost0_vbusvldextsel USB HOST0 external VBUS valid select This signal selects the VBUSVLDEXT input or the internal Session Valid comparator to indicate when the VBUS signal on the USB cable is valid. 1'b1: The VBUSVLDEXT input is used. 1'b0: The internal Session Valid comparator is used.
0	RW	0x0	usbhost0_vbusvldext USB HOST0 external VBUS valid indicator This signal is valid in Device mode and only when the VBUSVLDEXTSEL signal is set to 1. VBUSVLDEXT indicates whether the VBUS signal on the USB cable is valid. In addition, BUSVLDEXT enables the pullup resistor on the D+ line. 1'b1: The VBUS signal is valid, and the pull-up resistor on D+ is enabled. 1'b0: The VBUS signal is not valid, and the pull-up resistor on D+ is disabled.

**GRF\_UOC1\_CON3**

Address: Operational Base + offset (0x0340)

UOC1 control register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15	RW	0x0	usbhost0_ohci_susp_lgcy USB HOST0 ohci_susp_lgcy bit control
14	RW	0x0	usbhost0_ohci_cntsel USB HOST0 ohci_cntsel bit control
13	RW	0x0	usbhost0_utmiotg_idpullup USB HOST0 idpullup bit control
12	RW	0x1	usbhost0_utmiotg_dppulldown USB HOST0 dppulldown bit control
11	RW	0x1	usbhost0_utmiotg_dmpulldown USB HOST0 dmpulldown bit control
10	RW	0x1	usbhost0_utmiotg_drvvbus USB HOST0 drvvbus bit contrl
9:7	RO	0x0	reserved
6	RW	0x1	usbhost0_ohci_clkcktrst USB HOST0 ohci_clkcktrst bit conrol
5	RW	0x0	usbhost0_utmi_termselect USB HOST0 utmi termination select 1'b1: full speed terminations are enabled 1'b0: high speed terminations are enabled
4:3	RW	0x0	usbhost0_utmi_xcvrselect USB HOST0 utmi transceiver select 2'b11: sends an LS packet on an FS bus or receives an LS packet 2'b10: LS transceiver 2'b01: FS transceiver 2'b00: HS transceiver

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:1	RW	0x0	usbhost0_utmi_opmode USB HOST0 utmi operation mode This controller bus selects the UTMI+ operation mode 2'b11: normal operation without SYNC or EOP generation 2'b10: disable bit stuffing and NRZI encoding 2'b01: no-driving 2'b00: normal
0	RW	0x1	usbhost0_utmi_suspend_n USB HOST0 suspend mode enable 1'b1: normal operation mode 1'b0: suspend mode

**GRF\_UOC1\_CON4**

Address: Operational Base + offset (0x0344)

UOC1 control register 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RW	0x1	usbhost0_incr4_en USB HOST0 incr4_en bit control
14	RW	0x1	usbhost0_incr16_en USB HOST0 incr16_en bit control
13	RW	0x0	usbhost0_hubsetup_min USB HOST0 hubsetup_min bit control
12	RW	0x0	usbhost0_app_start_clk USB HOST0 app_start_clk bit control
11:6	RW	0x20	usbhost0_fadj_val_common USB HOST0 fadj_val_common bit control
5:0	RW	0x20	usbhost0_fadj USB HOST0 fadj bit control

**GRF\_UOC2\_CON0**

Address: Operational Base + offset (0x0348)

UOC2 control register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15	RW	0x0	usbhost1_linestate_irq_pd USB HOST1 linestate interrupt pending bit
14	RW	0x0	usbhost1_linestate_irq_en USB HOST1 line state interrupt enable
13	RW	0x0	<p>usbhost1_siddq</p> <p>USB HOST1 IDDQ test enable</p> <p>This test signal enables you to perform IDDQ testing by powering down all analog blocks.</p> <p>1'b1: The analog blocks are powered down.</p> <p>1'b0: The analog blocks are powered up.</p>
12	RW	0x0	<p>usbhost1_port_reset</p> <p>USB HOST1 per-port reset</p> <p>When asserted, this customer-specific signal resets the corresponding port transmit and receive logic without disabling the clocks within the PHY.</p> <p>1'b1: The transmit and receive finite state machines (FSMs) are reset, and the line_state logic combinatorially reflects the state of the single-ended receivers.</p> <p>1'b0: The transmit and receive FSMs are operational, and the line_state logic becomes sequential after 11 PHYCLOCK cycles.</p>
11:8	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:5	RW	0x4	<p>usbhost1_tune USB HOST1 VBUS valid threshold adjustment This bus adjusts the voltage level for the VBUS Valid threshold.</p> <p>3'b111: +9% 3'b110: +6% 3'b101: +3% 3'b100: Design default 3'b011: -3% 3'b010: -6% 3'b001: -9% 3'b000: -12%</p>
4	RW	0x0	<p>usbhost1_disable USB HOST1 block disable 1'b1: the USB HOST1 block is power down 1'b0: the USB HOST1 block is power up</p>
3:1	RW	0x4	<p>usbhost1_compdistune USB HOST1 disconnect threshold adjustment This bus adjusts the voltage level for the threshold used to detect a disconnect event at the host.</p> <p>3'b111: +4.5% 3'b110: +3% 3'b101: +1.5% 3'b100: Design default 3'b011: -1.5% 3'b010: -3% 3'b001: -4.5% 3'b000: -6%</p>
0	RW	0x1	<p>usbhost1_common_on_n USB HOST1 common block power-down control This signal controls the power-down signals in the XO, Bias, and PLL blocks when the USB 2.0 PHY is in Suspend or Sleep mode.</p> <p>1'b1: In Suspend mode, the XO, Bias, and PLL blocks are powered down. In Sleep mode, the Bias and PLL blocks are powered down.</p> <p>1'b0: In Suspend mode, the XO, Bias, and PLL blocks remain powered in Suspend mode. In Sleep mode, if the reference clock is a crystal, the XO block remains powered.</p>

**GRF\_UOC2\_CON1**

Address: Operational Base + offset (0x034c)

UOC2 control register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:14	RW	0x1	<p>usbhost1_txrisetune</p> <p>USB HOST1 HS transmitter rise/fall time adjustment</p> <p>This bus adjusts the rise/fall times of the high-speed waveform.</p> <p>2'b11: -20%</p> <p>2'b10: -15%</p> <p>2'b01: design default</p> <p>2'b00: +10%</p>
13:12	RW	0x3	<p>usbhost1_txhsxvtune</p> <p>USB HOST1 transmitter high-speed crossover adjustment</p> <p>This bus adjusts the voltage at which the DP and DM signals cross while transmitting in HS mode.</p> <p>2'b11: Default setting</p> <p>2'b10: +15 mV</p> <p>2'b01: -15 mV</p> <p>2'b00: Reserved</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:8	RW	0x3	<p>usbhost1_txvreftune            USB HOST1 HS DC voltage level adjustment            This bus adjusts the high-speed DC level voltage.</p> <p>4'b1111: +8.75%            4'b1110: +7.5%            4'b1101: +6.25%            4'b1100: +5%            4'b1011: +3.75%            4'b1010: +2.5%            4'b1001: +1.25%            4'b1000: Design default            4'b0111: -1.25%            4'b0110: -2.5%            4'b0101: -3.75%            4'b0100: -5%            4'b0011: -6.25%            4'b0010: -7.5%            4'b0001: -8.75%            4'b0000: -10%</p>
7:4	RW	0x3	<p>usbhost1_txfslstune            USB HOST1 FS/LS source impedance adjustment            This bus adjusts the low- and full-speed single-ended source impedance while driving high. The following adjustment values are based on nominal process, voltage, and temperature.</p> <p>4'b1111: -5%            4'b0111: -2.5%            4'b0011: Design default            4'b0001: +2.5%            4'b0000: +5%</p>
3	RW	0x0	<p>usbhost1_txpreemppulse<sup>tune</sup>            USB HOST1 HS transmitter pre-emphasis duration control            This signal controls the duration for which the HS pre-emphasis current is sourced onto DP0 or DM0. transition in HS mode.</p> <p>1'b1: 1X, short pre-emphasis current duration            1'b0: (desian default) 2X, long pre-emphasis currrent duration</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:0	RW	0x3	usbhost1_sqrxtune USB HOST1 squelch threshold adjustment This bus adjusts the voltage level for the threshold used to detect valid high-speed data. 3'b111: -20% 3'b110: -15% 3'b101: -10% 3'b100: -5% 3'b011: Design default 3'b010: +5% 3'b001: +10% 3'b000: +15%

**GRF\_UOC2\_CON2**

Address: Operational Base + offset (0x0350)

UOC2 control register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15	RW	0x0	usbhost1_acaenb USB HOST1 ACA ID_OTG pin resistance detection enable 1'b1: enable detection on resistance on the ID_OTG pin of an ACA 1'b0: disable detection on resistance on the ID_OTG pin of an ACA

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14	RW	0x0	usbhost1_dcdenb USB HOST1 data contact detection enable 1'b1: IDP_SRC current is sourced onto DP, pull-down resistance on DMA is enabled 1'b0: IDP_SRC current is disable, pull-down resistance on DM is disabled
13	RO	0x0	reserved
12:11	RW	0x1	usbhost1_txrestune USB HOST1 source impedance adjustment 2'b11: source impedance is decreased by 4ohm 2'b10: source impedance is decreased by 2ohm 2'b01: design default 2'b00: source impedance is decreased by 1.5ohm
10	RW	0x1	usbhost1_sleepm USB HOST1 sleep mode enable Asserting this signal place the USB PHY in sleep mode. 1'b0: sleep mode enable 1'b1: normal mode
9	RO	0x0	reserved
8	RW	0x1	usbhost1_retenable_n USB HOST1 retention mode enable 0: retention mode enable 1: retention mode disable
7	RW	0x0	usbhost1_vdatsrcenb USB HOST1 battery charging sourcing select 1'b1: data source voltage is enable 1'b0: data source voltage is disable
6	RW	0x0	usbhost1_vdatdetenb USB HOST1 battery charging attach/connect detection enable 1'b1: enable 1'b0: disable
5	RW	0x0	usbhost1_chrgsel USB HOST1 battery charging source select 1'b1: data source voltage is sourced onto DM and sunk from DP 1'b0: data source voltage is sourced onto DP and sunk from DM

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:3	RW	0x1	usbhost1_txpreempampktune 2'b11: 3X pre-emphasis current 2'b10: 2X pre-emphasis current 2'b01: 1X pre-emphasis current 2'b00: HS Transmitter Pre-Emphasis is disabled
2	RW	0x0	usbhost1_soft_con_sel 1'b0: software control usb host1 disable 1'b1: software control usb host1 enable
1	RW	0x0	usbhost1_vbusvldexsel USB HOST1 external VBUS valid select This signal selects the VBUSVLDEXT input or the internal Session Valid comparator to indicate when the VBUS signal on the USB cable is valid. 1'b1: The VBUSVLDEXT input is used. 1'b0: The internal Session Valid comparator is used.
0	RW	0x0	usbhost1_vbusvldext USB HOST1 external VBUS valid indicator This signal is valid in Device mode and only when the VBUSVLDEXTSEL signal is set to 1. VBUSVLDEXT indicates whether the VBUS signal on the USB cable is valid. In addition, BUSVLDEXT enables the pullup resistor on the D+ line. 1'b1: The VBUS signal is valid, and the pull-up resistor on D+ is enabled. 1'b0: The VBUS signal is not valid, and the pull-up resistor on D+ is disabled.

**GRF\_UOC2\_CON3**

Address: Operational Base + offset (0x0354)

UOC2 control register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:14	RW	0x0	usbhost1_scaledown_mode USB HOST1 scale down mode control
13	RW	0x0	usbhost1_utmiotg_idpullup USB HOST1 idpullup bit control
12	RW	0x1	usbhost1_utmiotg_dppulldown USB HOST1 dppulldown bit control
11	RW	0x1	usbhost1_utmiotg_dmpulldown USB HOST1 dmpulldown bit control
10	RW	0x1	usbhost1_utmiotg_drvvbus USB HOST1 drvvbus bit contrl
9:6	RO	0x0	reserved
5	RW	0x0	usbhost1_utmi_termselect USB HOST1 utmi termination select 1'b1: full speed terminations are enabled 1'b0: high speed terminations are enabled
4:3	RW	0x0	usbhost1_utmi_xcvrselect USB HOST1 utmi transceiver select 2'b11: sends an LS packet on an FS bus or receives an LS packet 2'b10: LS transceiver 2'b01: FS transceiver 2'b00: HS transceiver

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:1	RW	0x0	<p>usbhost1_utmi_opmode USB HOST1 utmi operation mode This controller bus selects the UTMI+ operation mode</p> <p>2'b11: normal operation without SYNC or EOP generation 2'b10: disable bit stuffing and NRZI encoding 2'b01: no-driving 2'b00: normal</p>
0	RW	0x1	<p>usbhost1_utmi_suspend_n USB HOST1 suspend mode enable</p> <p>1'b1: normal operation mode 1'b0: suspend mode</p>

**GRF\_UOC3\_CON0**

Address: Operational Base + offset (0x0358)

UOC3 control register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;</p>
15	RO	0x0	reserved
14	RW	0x0	hsicphy_soft_con_sel HSIC PHY software control enable
13	RW	0x1	hsicphy_txbitstuffen HSIC high byte transmit bit-stuffing enable this controller signal controls by stuffing on DATAIN[15:8] when OPMODE[1:0]=2'b11 1'b1: bit stuffing is enabled 1'b0: bit stuffing is disabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x1	hsicphy_txbitstuffen HSIC low byte transmit bit-stuffing enable this controller signal controls biy stuffing on DATAIN[7:0] when OPMODE[1:0]=2'b11 1'b1: bit stuffing is enabled 1'b0: bit stuffing is disabled
11	RW	0x0	hsichhy_siddq HSIC SIDDQ test enable 1'b1: the analog blocks are power down 1'b0: the analog blocks are power up
10	RW	0x0	hsicphy_port_reset HSIC per-port reset when asserted, this customer-specific signal reset the corresponding prot's transmit and receive logic without disabling the clocks within the HSIC PHY 1'b1: the transmit and receive FSMs are reset 1'b0: tjhe transmit and receive FSMs are operational
9:6	RW	0x3	hsicphy_txsrtune drive slew rate adjustment 4'b1111: +20% 4'b0111: +10% 4'b0011: design default 4'b0001: -10% 4'b0000: -20%
5:4	RW	0x2	hsicphy_txrpdtune HSIC driver pull-down impedance adjustment 2'b11: -5% 2'b10: design default 2'b01: +5% 2'b00: +11%
3:2	RW	0x2	hsicphy_txrputune HSIC driver pull-up impedance adjustment 2'b11: -5% 2'b10: design default 2'b01: +5% 2'b00: +11%
1	RW	0x1	hsicphy_dmpulldown HSIC bus keepers resistor enable This control signal selects the HSIC PHY to operate as a host or deivce.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x1	hsicphy_dppulldown HSIC bus keepers resistor enable This control signal detects that the HSIC PHY is being used as a host.

**GRF\_UOC3\_CON1**

Address: Operational Base + offset (0x035c)

UOC3 control register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:5	RO	0x0	reserved
4	RW	0x0	hsic_utmi_xcvrselect HSIC PHY transceiver select 1'b1: transceiver is in suspend, resume or connect mode 1'b0: transceiver is in HS mode
3:2	RW	0x0	hsic_utmi_opmode HSIC PHY operation mode 2'b11: normal mode without SYNC or EOP generation 2'b10: disable bit stuffing and NRZI encodeing 2'b01: Non-driving 2'b00: normal
1	RW	0x1	hsic_utmi_suspend_n HSIC PHY suspend mode enable Asserting this signal places the HSIC PHY in suspend mode. 1'b1: normal mode 1'b0: suspend mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x1	<p>hsic_utmi_sleep_n HSIC PHY sleep mode enable Asserting this signal places the HSIC PHY in sleep mode. 1'b1: normal mode 1'b0: sleep mode</p>

**GRF\_UOC4\_CON0**

Address: Operational Base + offset (0x0360)

UOC4 control register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;</p>
15:14	RW	0x0	<p>drvbus_out_sel0 USB PHY drv vbus output select 0 2'b00: USB OTG drv vbus 2'b01: USB HOST0 drv vbus 2'b1x: USB HOST1 drv vbus</p>
13:12	RW	0x1	<p>drvbus_out_sel1 USB PHY drv vbus output select 1 2'b00: USB OTG drv vbus 2'b01: USB HOST0 drv vbus 2'b1x: USB HOST1 drv vbus</p>
11:10	RO	0x0	reserved
9	RW	0x0	<p>hsic_app_prt_ovrcur HSIC app_prt_ovrcur bit control</p>
8	RW	0x0	<p>hsic_autoppd_on_overcur HSIC autoppd_on_overcur bit control</p>
7	RW	0x1	<p>hsic_word_if HSIC word_if bit control</p>
6	RW	0x0	<p>hsic_sim_mode HSIC sim_mode bit control</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	hsic_incrx_en HSIC incr <sub>x</sub> _en bit control
4	RW	0x0	hsic_incr8_en HSIC incr <sub>8</sub> _en bit control
3	RW	0x0	hsic_incr4_en HSIC incr <sub>4</sub> _en bit control
2	RW	0x0	hsic_incr16_en HSIC incr <sub>16</sub> _en bit control
1	RW	0x0	hsic_hubsetup_min HSIC hubsetup_min bit control
0	RW	0x0	hsic_app_start_clk HSIC app_start_clk bit control

**GRF\_UOC4\_CON1**

Address: Operational Base + offset (0x0364)

UOC4 control register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:12	RO	0x0	reserved
11:6	RW	0x20	hsic_fladj_val_common HSIC fladj_val_common bit control
5:0	RW	0x20	hsic_fladj HSIC fladj bit control

**GRF\_PVTM\_CON0**

Address: Operational Base + offset (0x0368)

PVT monitor control register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;</p>
15:10	RO	0x0	reserved
9	RW	0x0	<p>pvtm_gpu_osc_en pd_gpu PVT monitor oscillator enable 1'b1: enable 1'b0: disable</p>
8	RW	0x0	<p>pvtm_gpu_start pd_gpu PVT monitor start control</p>
7:2	RO	0x0	reserved
1	RW	0x0	<p>pvtm_core_osc_en pd_core PVT monitor oscillator enable 1'b1: enable 1'b0: disable</p>
0	RW	0x0	<p>pvtm_core_start pd_core PVT monitor start control</p>

**GRF\_PVTM\_CON1**

Address: Operational Base + offset (0x036c)

PVT monitor control register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x016e3600	<p>pvtm_core_cal_cnt pd_core pvtm calculator counter</p>

**GRF\_PVTM\_CON2**

Address: Operational Base + offset (0x0370)

PVT monitor control register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x016e3600	<p>pvtm_gpu_cal_cnt pd_gpu pvtm calculator counter</p>

**GRF\_PVTM\_STATUS0**

Address: Operational Base + offset (0x0374)

PVT monitor status register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	pvtm_core_freq_done pd_core pvtm frequency calculate done status
0	RW	0x0	pvtm_gpu_freq_done pd_gpu pvtm frequency calculate done status

### GRF\_PVTM\_STATUS1

Address: Operational Base + offset (0x0378)

PVT monitor status register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	pvtm_core_freq_cnt pd_core pvtm frequency count

### GRF\_PVTM\_STATUS2

Address: Operational Base + offset (0x037c)

PVT monitor status register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	pvtm_gpu_freq_cnt pd_gpu pvtm frequency count

### GRF\_IO\_VSEL

Address: Operational Base + offset (0x0380)

IO voltage select

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:10	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	gpio1830_v18sel GPIO1830 IO domain 1.8V voltage selection 1'b0: 3.3V 1'b1: 1.8V
8	RW	0x0	gpio30_v18sel GPIO30 IO domain 1.8V voltage selection 1'b0: 3.3V 1'b1: 1.8V
7	RW	0x0	sdcard_v18sel SDCARD IO domain 1.8V voltage selection 1'b0: 3.3V 1'b1: 1.8V
6	RW	0x0	audio_v18sel AUDIO IO domain 1.8V voltage selection 1'b0: 3.3V 1'b1: 1.8V
5	RW	0x0	bb_v18sel BB IO domain 1.8V voltage selection 1'b0: 3.3V 1'b1: 1.8V
4	RW	0x0	wifi_v18sel WIFI IO domain 1.8V voltage selection 1'b0: 3.3V 1'b1: 1.8V
3	RW	0x0	flash1_v18sel FLASH1 IO domain 1.8V voltage selection 1'b0: 3.3V 1'b1: 1.8V
2	RW	0x1	flash0_v18sel FLASH0 IO domain 1.8V voltage selection 1'b0: 3.3V 1'b1: 1.8V
1	RW	0x0	dvp_v18sel DVP IO domain 1.8V voltage selection 1'b0: 3.3V 1'b1: 1.8V
0	RW	0x0	lc当地_v18sel LCDC IO domain 1.8V voltage selection 1'b0: 3.3V 1'b1: 1.8V

**GRF\_SARADC\_TESTBIT**

Address: Operational Base + offset (0x0384)

SARADC Test bit register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:8	RO	0x0	reserved
7:0	RW	0x00	saradc_testbit SARADC test bit

**GRF\_TSADC\_TESTBIT\_L**

Address: Operational Base + offset (0x0388)

TSADC Test bit low register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x0000	tsadc_testbit_l Low 16bits of TSADC test bit

**GRF\_TSADC\_TESTBIT\_H**

Address: Operational Base + offset (0x038c)

TSADC Test bit high register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;</p>
15:0	RW	0x0000	tsadc_testbit_h High 16bits of TSADC test bit

**GRF\_OS\_REG0**

Address: Operational Base + offset (0x0390)

OS register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	os_reg0 OS register 0

**GRF\_OS\_REG1**

Address: Operational Base + offset (0x0394)

OS register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	os_reg1 OS register 1

**GRF\_OS\_REG2**

Address: Operational Base + offset (0x0398)

OS register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	os_reg2 OS register 2

**GRF\_OS\_REG3**

Address: Operational Base + offset (0x039c)

OS register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	os_reg3 OS register 3

**GRF\_SOC\_CON15**

Address: Operational Base + offset (0x03a4)

SoC control register 15

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>write_enable</p> <p>bit0~15 write enable</p> <p>When bit 16=1, bit 0 can be written by software .</p> <p>When bit 16=0, bit 0 cannot be written by software;</p> <p>When bit 17=1, bit 1 can be written by software .</p> <p>When bit 17=0, bit 1 cannot be written by software;</p> <p>.....</p> <p>When bit 31=1, bit 15 can be written by software .</p> <p>When bit 31=0, bit 15 cannot be written by software;</p>
15	RW	0x0	<p>grf_dclk1_lvds_inv_sel</p> <p>Inversion of VOP_LIT dclk for LVDS selection</p> <p>1'b1: invert</p> <p>1'b0: not invert</p>
14	RW	0x0	<p>grf_dclk1_lvds_div2_sel</p> <p>2 divide frequency of VOP_LIT dclk for LVDS selection</p> <p>1'b1: 2 divide frequency</p> <p>1'b0: no divide frequency</p>
13	RW	0x0	<p>grf_dclk0_lvds_inv_sel</p> <p>Inversion of VOP_BIG dclk for LVDS selection</p> <p>1'b1: invert</p> <p>1'b0: not invert</p>
12	RW	0x0	<p>grf_dclk0_lvds_div2_sel</p> <p>2 divide frequency of VOP_BIG dclk for LVDS selection</p> <p>1'b1: 2 divide frequency</p> <p>1'b0: no divide frequency</p>
11	RO	0x0	reserved
10:8	RW	0x0	<p>dphy_tx0_turnrequest</p> <p>MIPI DPHY TX0 turn around request</p> <p>Every bit for one lane, bit2 is for lane3, bit1 is for lane2, bit0 is for lane1.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:4	RW	0x0	dphy_tx1rx1_turnrequest MIPI DPHY TX1RX1 turn around request Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.
3:0	RW	0x0	dphy_rx0_turnrequest MIPI DPHY RX0 turn around request Every bit for one lane, bit3 is for lane3, bit2 is for lane2, bit1 is for lane1, bit0 is for lane0.

**GRF\_SOC\_CON16**

Address: Operational Base + offset (0x03a8)

SoC control register 16

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software . When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software . When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software . When bit 31=0, bit 15 cannot be written by software;
15:2	RO	0x0	reserved
1	RW	0x0	grf_con_dsi1_dpiupdatecfg DSI host1 dpiupdatecfg bit control
0	RW	0x0	grf_con_dsi0_dpiupdatecfg DSI host0 dpiupdatecfg bit control

## Chapter 8 Core System

### 8.1 Overview

The Core system of the device is based on the symmetric multiprocessor (SMP) architecture, contain quad Cortex-A17. The Cortex-A17 implements 32KB L1 instruction cache, 32KB L1 data cache, 1MB L2 cache. It delivers higher performance and optimal power management, debug and emulation capabilities.

The core system also contain a PL301 interconnect, which connect the A17's peripheral port and the generic interrupt controller (GIC400).

The core system contain the ARM's coresight cell , which helps to do debug and trace .

The interrupt controller is also included in this system, for detail description , please reference to chapter 12.

The Core system supports following features:

- ARM Coretex-A17 based quad MPU subsystem with SMP architecture
  - Cortex-A17 core revision r0p1
  - Full implements the ARMv7-A architecture profile that includes SIMDv2 and VFPv4-D32 extensions
  - 32KB L1 I-cache and 32KB L1 D-cache with 64-byte line size and 4-way set associative per CPU
  - A 32-entry,fully associative, instruction micro TLB.
  - A 32-entry,fully associative, data micro TLB.
  - A 1024-entry, 4-way, unified main TLB with hit-under-miss capabilities.
  - Memory management unit (MMU)
  - Automatic cache coherency between L1 data caches within the cluster.
  - Interrupt controller with 128 hardware interrupt inputs.
  - 1MB L2 cache with 64-byte line size, and 16-way set associative.
- standard CoreSight™ components to support SMP debug and emulation
  - Program trace macrocell (PTM)
  - Emulation logic (cross-triggers)
  - TPIU for trace.

### 8.2 Block Diagram

The Core system comprises with:

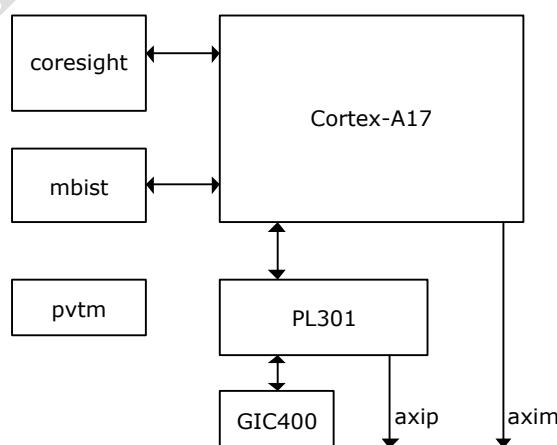


Fig. 8-1 Block Diagram

### 8.3 Function Description

Please refer to the document cortex\_a17\_r0p0\_trm.pdf for the cpu detail description.

Rockchip Confidential

## Chapter 9 Interconnect

### 9.1 Overview

The chip-level interconnect consists of main interconnect and peri interconnects. It enables communication among the modules and subsystems in the device.

The main interconnect supports the following features:

- Cross-bar exchange network
- A special internal slave for accessing the configuration register
- Little-endian platform
- Embedded memory scheduler for ddr transaction generation
- QoS management for optimizing the transaction flow
- Transaction statistics for analyzing the transaction flow
- Security protection mechanism to compatible with the TrustZone technology
- The peri interconnect belong to peripheral system which is responsible for peripheral devices control such as usb device, flash device, uart, spi etc.

### 9.2 Block Diagram

The interconnect comprises with:

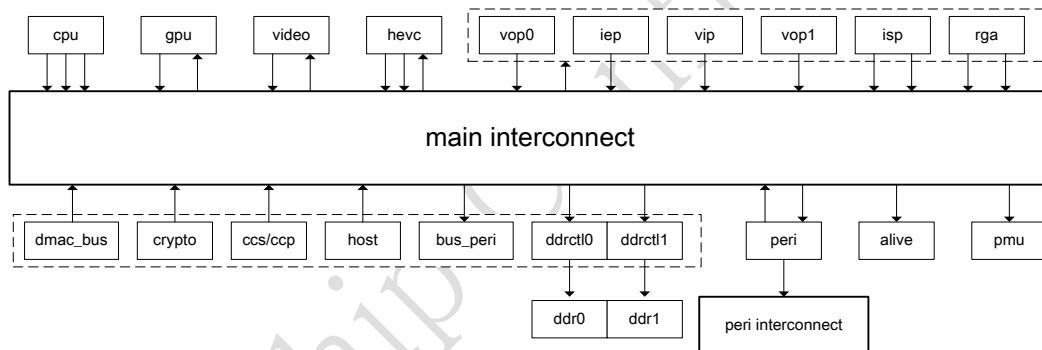


Fig. 9-1 Block Diagram

### 9.3 Function Description(main interconnect)

#### 9.3.1 Master & Slave

The main interconnect is connected with all the related IPs of the system, the interface between the IP and the interconnect is called as NIU(native interface unit). All the connected NIU are list as bellowing:

Table 9-1 Master NIU

Master NIU	Description
cpup	Cortex-A17 AXI P port, access to any peripheral device
cpum	Cortex-A17 AXI M port , access to

main memory	
dbg	Jtag Debug master , access to any peripheral
bus_dmac	DMAC in pd_bus power domain
crypto	Crypto , used in trustzone technolgy
ccs	CCS, used in trustzone technolgy
ccp	CCP, used in trustzone technolgy
gpu	Mali-T760 of pd_gpu power domain
hevc	HEVC of pd_hevc power domain
video	Video codec of pd_video power domain
vop_big	Lcdc full function of pd_vio power domain
iep	IEP of pd_vio power domain
vip	VIP of pd_vio power domain
vop_lit	Lcdc lite of pd_vio power domain
isp	ISP of pd_vio power domain
rga	RGA of pd_vio power domain
peri	Peri-PL301 of pd_peri power domain

Table 9-2 slave NIU

Slave NIU	Description
msch0	Memory scheduler channel-0 in pd_bus power domain
msch1	Memory scheduler channel-1 in pd_bus power domain
bus_ahb	Ahb slave in pd_bus power domain
imem	Internal memory in pd_bus power domain

bus_apb	Apb slave in pd_bus power domain
pmu_apb	Pmu apb slave in pd_pmu power domain
alive_apb	Alive apb slave in pd_alive power domain
vio_ahb	Vio ahb slave in pd_vio power domain
gevc_ahb	Hevc ahb slave in pd_hevc power domain
video_ahb	Video ahb slave in pd_video power domain
gpu_axi	Gpu axi slave in pd_gpu power domain
service_bus	Service module inside the interconnect in pd_bus power domain, used as register configuration
service_core	Service module in pd_core power domain
service_dmac	Service module in pd_bus power domain
service_gpu	Service module in pd_gpu power domain
service_hevc	Service module in pd_hevc power domain
service_peri	Service module in pd_peri power domain
service_vio	Service module in pd_vio power domain
service_video	Service module in pd_video power domain

### 9.3.2 Clock & Power domain

The interconnect is divided to several clock domain and power domain. Each domain contain different IPs.

The clock/power domain and IP combination is list as bellowing:

Table 9-3 Clock and Power domain

Clock domain	Clock	Power domain	IP
alive_clk_dm	alive_pclk	alive_pwr	CRU/PLL/GRF/GPIO/TIMER/WDT

bus_clk_dm	bus_aclk	bus_pwr	IMEM
	bus_hclk		I2S/SPDIF/CRYPTO
	bus_pclk		TIMER/PWM/I2C/UART/DMAC/PCTL
core_clk_dm	cpum_aclk/ cpup_aclk	core_pwr	Cortex-A17
gpu_clk_dm	gpu_aclk	gpu_pwr	Mali-T760
hevc_aclk_dm	hevc_aclk	hevc_pwr	HEVC
hevc_hclk_dm	hevc_hclk		
msch0_clk_dm	msch0_clk	bus_pwr	Memory scheduler 0
msch1_clk_dm	msch1_clk		Memory scheduler 1
peri_clk_dm	peri_aclk	peri_pwr	Peri-PL301
pmu_clk_dm	pmu_pclk	pmu_pwr	PMU/PMU_IMEM/SGRF/GPIO
video_aclk_dm	video_aclk	video_pwr	VCODEC
video_hclk_dm	video_hclk		
vio0_clk_dm	vio0_aclk	vio_pwr	VOP_BIG/IEP/VIP
vio01_clk_dm	vio1_aclk		ISP/VOP_LIT
vio2_clk_dm	vio2_aclk		RGA
vio_hclk_dm	vio_hclk		

### 9.3.3 QoS management

The interconnect offers 4 modes of qos management:

- None: QoSGenerator is disabled, and priority information are stuck at 0.
- Fixed: QoSGenerator drives applies a fixed urgency to read transactions, and a (possibly different) urgency to write transactions.
- Limiter: QoSGenerator behaves as in fixed mode, but limits the traffic bandwidth coming from that socket, possibly stalling requests if the initiator attempts to exceed its budget.
- Regulator: QoSGenerator promotes or demotes hurry, depending the bandwidth obtained by the initiator is below or beyond a bandwidth budget. As transactions exceeding the bandwidth limit are sent (even though demoted), the regulator mode may be considered as a softer version of the limiter mode.

## Limiter Behavior

When configured in bandwidth limiter, the unit uses a 23 bit counter to measure the average bandwidth. This counter has a 1/256 byte resolution and works as follows:

- Adds the number of byte rounded up to 16 (1 -> 16) and then multiplied by 256 to the current value, each time a request is sent.
- Subtracts the Bandwidth register value every cycle. If the Counter becomes negative, force it to 0.
- If the Counter value is greater than the Saturation register value multiplied by 16\*256, any incoming request is stalled until this condition disappears. Note that the Counter cannot wrap-around because the maximum value it can reach is:  $\text{SaturationMax} \times 16 \times 256 + \text{BurstMax} \times 256 = 1023 \times 4K + 4K \times 256 = 5116K$  or  $223 = 8192K$ .

The following example will show the Counter behavior: 32 byte bursts, F=400MHz, BW=200MB/s, T=0.32us. The Bandwidth register will be set to  $256 \times 200 / 400 = 128$ , and the Saturation register to  $128 \times 0.32 \times 400 / 4096 = 4$  (which corresponds to 64 bytes).

## Regulator Behavior

When configured in bandwidth regulator, the unit uses a 23 bit counter to measure the average bandwidth. This counter has a 1/256 byte resolution and works as follows:

- Adds the number of byte rounded up to 16 and then multiplied by 256 to the current value, each time a response is received. If the result is greater than the Saturation register value multiplied by 16\*256, saturation to this value is applied.
- Subtracts the Bandwidth register value every cycle. If the Counter becomes negative, force it to 0.
- If the Counter value is less than or equal to the Saturation register value multiplied by  $16 \times 256 / 2$ , the SocketMst Hurry signal will be set to the HurryHigh register, and HurryLow otherwise. Note that Urgency and Press will be also set to the same value.

The following example will show the Counter behavior: 1Kbyte bursts, F=500MHz, BW=2GB/s, T=2.048us. The Bandwidth register will be set to  $256 \times 2000 / 500 = 1024$ , and the Saturation register to  $1024 \times 2.048 \times 500 / 4096 = 256$  (which corresponds to 4 Kbytes).

## QoS Generator Programming

Bandwidth: This  $\log_2(\text{socket.wData}/8) + 8$  bits register defines the bandwidth in 1/256th byte per cycle unit. This allows a 2 MByte/s resolution at 500MHz. When the bandwidth is given in MByte/s, the value of this register will

be equal to  $256 \times \text{BWMB/s} / \text{FMHz}$ .

Saturation: This 10 bits register defines the number of bytes used for bandwidth measurement. It is expressed in 16bytes unit (up to 16 Kbyte). Usually the integration window is given in us or in cycle: the value of this register will be equal to Bandwidth\*Tus\*MHz/(256\*16) or Bandwidth\*Ncycle/(256\*16).

### 9.3.4 Memory Scheduler

Memory scheduler is a special NIU of the interconnect, it mainly deal with the transaction inside the interconnect and convert it to the transaction which the ddr protocol controller can recognize.

There are two memory schedulers in the interconnect, each is in different clock domain. These two memory schedulers are totally equal in function .

Following table shows the software configurable setting for the memory scheduler when the system connected to different size of ddr device.

The DDRCONF[3:0] is a configurable register inside the interconnect.

R: indicates Row bits

B: indicates Bank bits

C: indicates Column bits

D: indicates Chip selects bits

Table 9-4 DDR configuration

<b>DDR CONF[3:0]</b>	
0	R DBBB RRRR RRRR RRRR RRRC CCCC CCCC C---
1	C RRRD RRRR RRRR RRRR RBBB CCCC CCCC C---
2	C RRDR RRRR RRRR RRRR RBBB CCCC CCCC C---
3	C RDRR RRRR RRRR RRRR RBBB CCCC CCCC C---
4	C DRRR RRRR RRRR RRRR RBBB CCCC CCCC C---
5	R RRDR RRRR RRRR RRRR BBBC CCCC CCCC C---
6	R RDRR RRRR RRRR RRRR BBBC CCCC CCCC C---
7	R DRDR RRRR RRRR RRRR BBBC CCCC CCCC C---
8	C CRDR DRDR RRRR RRRR RRBB BCCC CCCC C---
9	C CRRD RRRR RRRR RRRR RRBB BCCC CCCC C---
10	C CRDR RRRR RRRR RRRR RRBB BCCC CCCC C---

11	C CBRR DRRR RRRR RRRR RRRB BCCC CCCC C---
12	C RBRR DRRR RRRR RRRR RRBB CCCC CCCC C---
13	B RRRR DRRR RRRR RRRR RBBC CCCC CCCC C---
14	C DRBB BRRR RRRR RRRR RRRR CCCC CCCC C---
15	D RRRR RRRR RRRR RRRR BBBC CCCC CCCC C---

When access to the two memory schedulers, the interconnect supports different stride between the two memory schedulers. Because of this function, the transaction can be interleaved send to the two channel ddr devices.

The stride can be 0Bytes, 128Bytes, 256Bytes, 512Bytes, 4kBBytes.

It support the following configuration:

Table 9-5 DDR Stride

DDR STRIDE[4:0]	Channel 0 Address range	Channel 1 Address range	Stride size	Total size
5'b0_0000	0x0--0x0fff_ffff	0x1000_0000--0x1fff_ffff	256MB	512MB
5'b0_0001	0x0--0x1fff_ffff	0x2000_0000--0x3fff_ffff	512MB	1GB
5'b0_0010	0x0--0x3fff_ffff	0x4000_0000--0x7fff_ffff	1GB	2GB
5'b0_0011	0x0--0x7fff_ffff	0x8000_0000--0xffff_ffff	2GB	4GB
5'b0_0100	0x0--0x3fff_ffff	0x0--0x3fff_ffff	128B	1GB
5'b0_0101	0x0--0x3fff_ffff	0x0--0x3fff_ffff	256B	1GB
5'b0_0110	0x0--0x3fff_ffff	0x0--0x3fff_ffff	512B	1GB
5'b0_0111	0x0--0x3fff_ffff	0x0--0x3fff_ffff	4KB	1GB
5'b0_1000	0x0--0x7fff_ffff	0x0--0x7fff_ffff	128B	2GB
5'b0_1001	0x0--0x7fff_ffff	0x0--0x7fff_ffff	256B	2GB
5'b0_1010	0x0--0x7fff_ffff	0x0--0x7fff_ffff	512B	2GB
5'b0_1011	0x0--0x7fff_ffff	0x0--0x7fff_ffff	4KB	2GB
5'b0_1100	0x0--0xffff_ffff	0x0--0xffff_ffff	128B	4GB
5'b0_1101	0x0--0xffff_ffff	0x0--0xffff_ffff	256B	4GB
5'b0_1110	0x0--0xffff_ffff	0x0--0xffff_ffff	512B	4GB
5'b0_1111	0x0--0xffff_ffff	0x0--0xffff_ffff	4KB	4GB

5'b1_0000	0x0-- 0x7fff_ffff	0x0-- 0x7fff_ffff	128B	3GB
	0x8000_0000-- 0xbfff_ffff	0x8000_0000-- 0xbfff_ffff	128B	
5'b1_0001	0x0-- 0x7fff_ffff	0x0-- 0x7fff_ffff	256B	3GB
	0x8000_0000-- 0xbfff_ffff	0x8000_0000-- 0xbfff_ffff	256B	
5'b1_0010	0x0-- 0x7fff_ffff	0x0-- 0x7fff_ffff	512B	3GB
	0x8000_0000-- 0xbfff_ffff	0x8000_0000-- 0xbfff_ffff	512B	
5'b1_0011	0x0-- 0x7fff_ffff	0x0-- 0x7fff_ffff	4KB	3GB
	0x8000_0000-- 0xbfff_ffff	0x8000_0000-- 0xbfff_ffff	4KB	
5'b1_0100	0x0-- 0x0fff_ffff	0x1000_0000-- 0x1fff_ffff	256MB	1GB
	0x2000_0000-- 0x3fff_ffff	0x2000_0000-- 0x3fff_ffff	128B	
5'b1_0101	0x0-- 0x1fff_ffff	0x2000_0000-- 0x3fff_ffff	512MB	2GB
	0x4000_0000-- 0x7fff_ffff	0x4000_0000-- 0x7fff_ffff	128B	
5'b1_0110	0x0--0xffff_ffff		4GB	4GB
5'b1_0111		0x0000_0000--0xffff_ffff	4GB	4GB
5'b1_1010	0x0-0xffff_ffff	0x1_0000_0000—0x1_ffff_ffff	4GB	8GB
5'b1_1011	0x0-0x1_ffff_ffff	0x0-0x1_ffff_ffff	128B	8GB

The following picture shows the address mapping from soc system to ddr device space, when configure the ddrstride=5'b01000 ,interleaved size as 128Bytes.

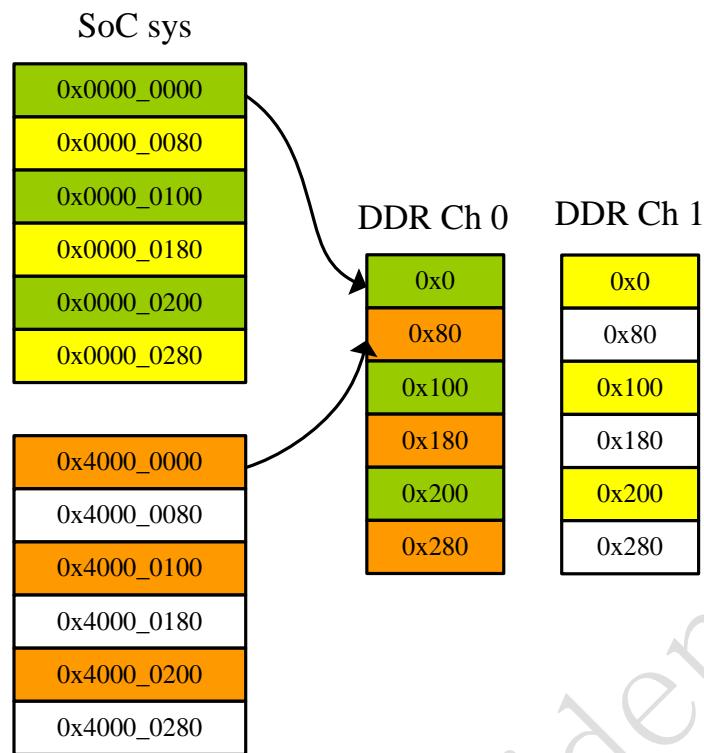


Fig. 9-2 DDR interleaved example

## 9.4 Register Description(main interconnect)

### 9.4.1 Internal Address Mapping

Table 9-6 Service module

Service Module	Base address
service_bus	0xffac_0000
service_core	0xffa8_0000
service_dmac	0xffa9_0000
service_gpu	0xffaa_0000
service_hevc	0xffaf_0000
service_peri	0xffab_0000
service_vio	0xffad_0000
service_video	0xffae_0000

Table 9-7 Service\_bus block

Register block inside service_bus	Internal offset
msch0	0x0
msch0_cpum_probe	0x400
msch0_gpu_probe	0x800
msch0_obsrv	0x180
msch0_peri_probe	0xc00
msch0_video_probe	0x1000
msch0_vio0_probe	0x1400
msch0_vio1_probe	0x1800
msch0_vio2_probe	0x1c00
msch1	0x80
msch1_cpum_probe	0x2000
msch1_gpu_probe	0x2400
msch1_obsrv	0x200
msch1_peri_probe	0x2800
msch1_video_probe	0x2c00
msch1_vio0_probe	0x3000
msch1_vio1_probe	0x3400
msch1_vio2_probe	0x3800

Table 9-8 Service\_core block

Register block inside service_core	Internal offset
cpum_r_qos	0x80
cpum_w_qos	0x100
cpup_qos	0x0

Table 9-9 Service\_dmac block

Register block inside service_dmac	Internal offset
bus_dmac_qos	0x0
ccp_qos	0x180
crypto_qos	0x100
ccs_qos	0x200
host_qos	0x80

Table 9-10 Service\_gpu block

Register block inside service_gpu	Internal offset
gpu_r_qos	0x0
gpu_w_qos	0x80

Table 9-11 Service\_hevc block

Register block inside service_hevc	Internal offset
hevc_r_qos	0x0
hevc_w_qos	0x100

Table 9-12 Service\_peri block

Register block inside service_peri	Internal offset
peri_qos	0x0

Table 9-13 Service\_vio block

Register block inside service_vio	Internal offset
vio0_iep_qos	0x500
vio0_vip_qos	0x480
vio0_vop_qos	0x400

vio1_isp_r_qos	0x900
vio1_isp_w0_qos	0x100
vio1_isp_w1_qos	0x180
vio1_vop_qos	0x0
vio2_rga_r_qos	0x800
vio2_rga_w_qos	0x880

#### 9.4.2 Registers Summary

##### service\_bus: memory scheduler channel-0 register summary

Name	Offset	Size	Reset Value	Description
coreid	0x0000	W	0x21501602	core id of memory scheduler 0
revisionid	0x0004	W	0x0126f200	revisionid
ddrconf	0x0008	W	0x00000000	ddr configuration register
ddrtiming	0x000c	W	0x2475931c	ddr timing register
ddrmode	0x0010	W	0x00000000	ddr mode register
readlatency	0x0014	W	0x00000032	read latency register
activate	0x0038	W	0x00000400	activate register
devtodev	0x003c	W	0x00000000	devtodev register

##### service\_core: cpup\_qos register summary

Name	Offset	Size	Reset Value	Description
priority	0x0008	W	0x00000101	priority register
mode	0x000c	W	0x00000003	qos mode register
bandwidth	0x0010	W	0x00000005	qos bandwidth register
saturation	0x0014	W	0x00000040	qos saturation register
extcontrol	0x0018	W	0x00000000	qos extcontrol register

Notes:**S**ize: **B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**-WORD (32 bits) access

service\_bus:msch1's register is same with msch0

other master's qos register is same with service\_core:cpup\_qos

### 9.4.3 Detail Register Description

#### **service\_bus: memory scheduler channel-0 coreid**

Address: Operational Base + offset (0x0000)

Bit	Attr	Reset Value	Description
31:0	RO	0x21501602	msch0' core id

#### **service\_bus: memory scheduler channel-0 revisionid**

Address: Operational Base + offset (0x0004)

Bit	Attr	Reset Value	Description
31:0	RO	0x0126f200	msch0' revision id

#### **service\_bus: memory scheduler channel-0 ddrconf**

Address: Operational Base + offset (0x0008)

Memory scheduler configuration register

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3:0	RW	0x0	DdrConf select the ddr rank,row,bank,col sequence

#### **service\_bus: memory scheduler channel-0 ddrtiming**

Address: Operational Base + offset (0x000c)

Memory scheduler timing register

Bit	Attr	Reset Value	Description
31	RW	0x0	BwRatio Bandwidth determines , in conjunction with field BwRatioExtended of register ddrmode, the number of DRAM data bus cycles required to process a scheduler data bus word.  When BwRatio and BwRatioExtended are set to 0, the bandwidth ratio is 1:1, that is one DRAM cycles per scheduler word.  When BwRatio is set to 1, and BwRatioExtended is set to 0, the bandwidth ratio is 2:1, that is ,two DRAM cycles per scheduler word.
30:26	RW	0x00	WrToRd Minimum time between the last DRAM Write command and a Read command.

Bit	Attr	Reset Value	Description
25:21	RW	0x00	RdToWr Minimum time between the last DRAM Read command and a Write command.
20:18	RW	0x0	BurstLen DRAM burst duration on the DRAM data bus. Also equal to minimum time between two DRAM commands.
17:12	RW	0x00	WrToMiss Minimum time between the last DRAM Write command and a new Read or Write command in another page of the same bank.
11:6	RW	0x00	RdToMiss Minimum time between the last DRAM Read command and a new Read or Write command in another page of the same bank.
5:0	RW	0x00	ActToAct Minimum time between two consecutive DRAM Activate commands on the same bank.

**service\_bus: memory scheduler channel-0 ddrmode**

Address: Operational Base + offset (0x0010)

Memory scheduler mode register

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	BwRatioExtended When 0, bandwidth ratio is determined by BwRatio. When 1, the bandwidth ratio is 4:1, that is four DRAM cycles per scheduler word, and BwRatio must be set to 0.
0	RW	0x0	AutoPrecharge When set to 1, pages are automatically closed after each access(no page hit). When set to 0, pages are left open until an access in a different page occurs.

**service\_bus: memory scheduler channel-0 readlatency**

Address: Operational Base + offset (0x0014)

Memory scheduler read latency register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
7:0	RW	0x32	ReadLatency Stores a user-defined read response latency ceiling, expressed in scheduler clock cycles. When the scheduler determines that the latency of a read transaction to the controller exceeded this value, the scheduler stalls further transactions to reduce the number of commands in the memory controller queue.

**service\_bus: memory scheduler channel-0 activate**

Address: Operational Base + offset (0x0038)

Memory scheduler activate register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved
10	RW	0x01	Fawbank Indicates the number of banks of a given device involved in the FAW period during which four banks can be activate.
9:4	RW	0x00	Faw The length of the FAW period, in cycles.
3:0	RW	0x00	Rrd The minimum number of scheduler clock cycles between two consecutive DRAM activate commands on different banks on the same device.

**service\_bus: memory scheduler channel-0 devtodev**

Address: Operational Base + offset (0x003c)

Memory scheduler devtodev register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:4	RW	0x00	Buswrtord The time delay, expressed in cycles, between a write and a read operation on different devices.
3:2	RW	0x00	Busrdtownr The time delay, expressed in cycles, between a read and a write operation on different devices.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RW	0x00	Busrdtord The time delay, expressed in cycles, between a read and a read operation on different devices.

Following is cpup port's QoS register detail description. Other ports have the same register. The only different is the base address's offset. The offset information is described in Table1-6 ~ Table1-13.

### **service\_core: cpup\_qos Priority**

Address: Operational Base + offset (0x0008)

CPU master0 priority register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:2	RW	0x2	P1 sets the high hurry level (i.e. when the measured bandwidth does not exceed the setting) when in Regulator mode, or read urgency level when in fixed or limiter mode.
1:0	RW	0x0	P0 sets the low hurry level, that is, when the measured bandwidth exceeds the setting, when in regulator mode, or write urgency level when in fixed or limiter mode.

### **service\_core: cpup\_qos Mode**

Address: Operational Base + offset (0x000c)

CPU master0 QoS mode register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1:0	RW	0x3	Mode determines which of the following modes the QoSGenerator will adopt at reset: 0 : None 1 : Fixed 2 : Limiter 3 : Regulator

### **service\_core: cpup\_qos Bandwidth**

Address: Operational Base + offset (0x0010)

CPU master0 QoS bandwidth register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved
10:0	RW	0x007	Bandwidth Parameter bandwidth determines the bandwidth triggering of the limiter or the regulator. It is expressed in bytes per second. This parameter becomes available when limiter or regulator hardware is implemented.

### **service\_core: cpup\_qos Saturation**

Address: Operational Base + offset (0x0014)

CPU master0 QoS saturation register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9:0	RW	0x3ff	Saturation Parameter saturation determines the excursion of the payload counter, used to estimate the bandwidth, expressed in bytes. This parameter becomes available when limiter or regulator hardware is implemented.

### **service\_core: cpup\_qos ExtControl**

Address: Operational Base + offset (0x0014)

CPU master0 QoS saturation register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
2	RW	0x0	Replace the External reference by the local clock.
1	RW	0x0	ExtThr input controls Low/High priority instead of bandwidth threshold.
0	W	0x0	Combines the Socket QoS with Regulator QoS

## **9.5 Function Description(peri interconnect)**

The peri interconnect contain the following master:

peri,gps,dmac,usb,nandc,mac,tsp,sdmmc.

A Global Programmers View (GPV) module exists to configure some properties of the

interconnect. The arbitration scheme of this interconnect is configurable through GPV.

The priority from high to low is as follow:

- peri/gps
- dmac
- usb
- nandc/mac

Customers can configure the Qos value through the GPV to change this priority. If you config them to same priority, then the interconnect uses a Least Recently Used (LRU) algorithm

## 9.6 Register Description(peri interconnect)

### 9.6.1 Registers Summary

Name	Offset	Size	Reset Value	Description
PERI_RQos_M0	0x42100	4bits	0x0003	peri AXI Port Read channel QoS value.
PERI_WQos_M0	0x42104	4bits	0x0003	peri AXI Port Write channel quality value.
PERI_RQos_M1	0x43100	4bits	0x0002	dmac Port Read channel QoS value.
PERI_WQos_M1	0x43104	4bits	0x0002	dmac Port Write channel quality value.
PERI_RQos_M2	0x44100	4bits	0x0001	nandc Port Read channel QoS value.
PERI_WQos_M2	0x44104	4bits	0x0001	nandc Port Write channel quality value.
PERI_RQos_M3	0x45100	4bits	0x0000	usb Port Read channel QoS value.
PERI_WQos_M3	0x45104	4bits	0x0000	usb Port Write channel quality value.
PERI_RQos_M4	0x46100	4bits	0x0000	gps Port Read channel QoS value.
PERI_WQos_M4	0x46104	4bits	0x0000	gps Port Write channel quality value.

Notes: Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

## 9.6.2 Detail Register Description

### **PERI\_RQos\_M0**

Address: Operational Base+0x42100

PERI\_AXI Port Read Qos register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x1	Read channel QoS value.Higher value indicates higher priority.

### **PERI\_WQos\_M0**

Address: Operational Base+0x42104

PERI\_AXI Port Write Qos register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x1	Write channel QoS value.Higher value indicates higher priority.

### **PERI\_RQos\_M1**

Address: Operational Base+0x43100

DMAC Port Read Qos register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	Read channel QoS value.Higher value indicates higher priority.

### **PERI\_WQos\_M1**

Address: Operational Base+0x43104

DMAC Port Write Qos register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	Write channel QoS value.Higher value indicates higher priority.

### **PERI\_RQos\_M2**

Address: Operational Base+0x44100

NANDC Port Read Qos register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	Read channel QoS value.Higher value indicates higher priority.

**PERI\_WQos\_M2**

Address: Operational Base+0x44104

NANDC Port Write Qos register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	Write channel QoS value.Higher value indicates higher priority.

**PERI\_RQos\_M3**

Address: Operational Base+0x45100

USB Port Read Qos register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	Read channel QoS value.Higher value indicates higher priority.

**PERI\_WQos\_M3**

Address: Operational Base+0x45104

USB Port Write Qos register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	Write channel QoS value.Higher value indicates higher priority.

**PERI\_RQos\_M4**

Address: Operational Base+0x46100

GPS Port Read Qos register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	Read channel QoS value.Higher value indicates higher priority.

**PERI\_WQos\_M4**

Address: Operational Base+0x46104

GPS Port Write Qos register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	Write channel QoS value.Higher value indicates higher priority.

## 9.7 Application Notes

### QoS setting

The cpum read channel, vop read channel , gpu write channel have the external QoS control.

After reset each master port both have priority setting as 1. It's recommended that field 0 of qos.ExtControl set to 1 to enable the external qos control. And priority setting of each master kept at 1.

### Idle request

The main interconnect supports flushing the ongoing transaction when the software needed to do so.

If the GPU power domain need to disconnect from the main interconnect, Idle request has to be sent to GPU NIU, the NIU will respond a ack, and when it's ready to be disconnect, one Idle signal will be send out . Then, if GPU still have transaction to be sent to the memory scheduler, it will be stalled by the NIU.

If the GPU system power domain is disconnected as the above flow, then CPU want to access to the GPU system, it will response error to CPU.

The sequence is like following figure shows:

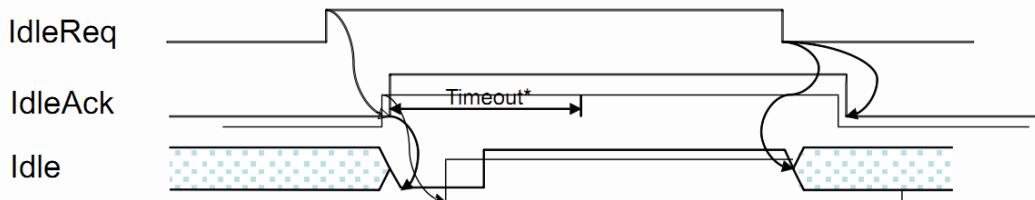


Fig. 9-3 Idle request

The idle request is set by PMU register.

## DDR timing examples

The following picture provides examples of DDR timing values, in clock cycles for register values, and nanoseconds for parameter values.

DDR type	DDR2-3E		DDR2-25E		DDR3-187E		DDR3-125E	
DDR frequency	333 MHz		400 MHz		533 MHz		800 MHz	
Scheduler frequency	333 MHz		400 MHz		533 MHz		400 MHz	
Burst length	4		8		8		8	
Register field	cycles	ns	cycles	ns	cycles	ns	cycles	ns
ActToAct	18	54	22	55	27	50.6	19	47.5
RdToMiss	9	27	9	22.5	15	28.1	11	27.5
WrToMiss	16	48	20	50	29	54.3	20	50
BurstLen	2	6	4	10	4	7.5	2	5
RdToWr	2	6	2	5	3	5.6	2	5
WrToRd	2	6	7	17.5	10	18.7	7	17.5
Rrd	4	12	4	10	6	11.2	3	7.5
Faw	17	51	18	45	27	50.6	16	24
BusRdToRd	1	3	1	2.5	1	1.8	1	2.5
BusRdToWr	2	6	2	5	2	3.7	2	5
BusWrToRd	2	6	2	5	2	3.7	2	5

Fig. 9-4 DDR timing example

## Chapter 10 DMA Controller for Bus System (DMAC\_BUS)

### 10.1 Overview

This device supports 2 Direct Memory Access (DMA) tops, one for cpu system (DMAC\_BUS), and the other one for Peripheral system (PERI\_DMAC).Both of these two dma support transfers between memory and memory, peripheral and memory.

DMAC\_BUS supports TrustZone technology and is under secure state after reset. The secure state can be changed by configuring TZPC module.

DMAC\_BUS is mainly used for data transfer of the following slaves: I2S0/I2S1/SPDIF/UART0/Embedded SRAM and transfer data from/to external DDR SDRAM.

Following table shows the DMAC\_BUS peripheral request mapping scheme.

Table 10-1 DMAC\_BUS Request Mapping Table

Req number	Source	Polarity
0	I2S tx	High level
1	I2S rx	High level
2	SPDIF	High level
3	SPDIF (8ch)	High level
4	UART DBG tx	High level
5	UART DBG rx	High level

DMAC\_BUS supports the following features:

- Supports Trustzone technology
- Supports 6 peripheral request
- Up to 64bits data size
- 5 channel at the same time
- Up to burst 16
- 10 interrupt output and 1 abort output
- Supports 32 MFIFO depth

### 10.2 Block Diagram

Figure 10-1 shows the block diagram of DMAC\_BUS

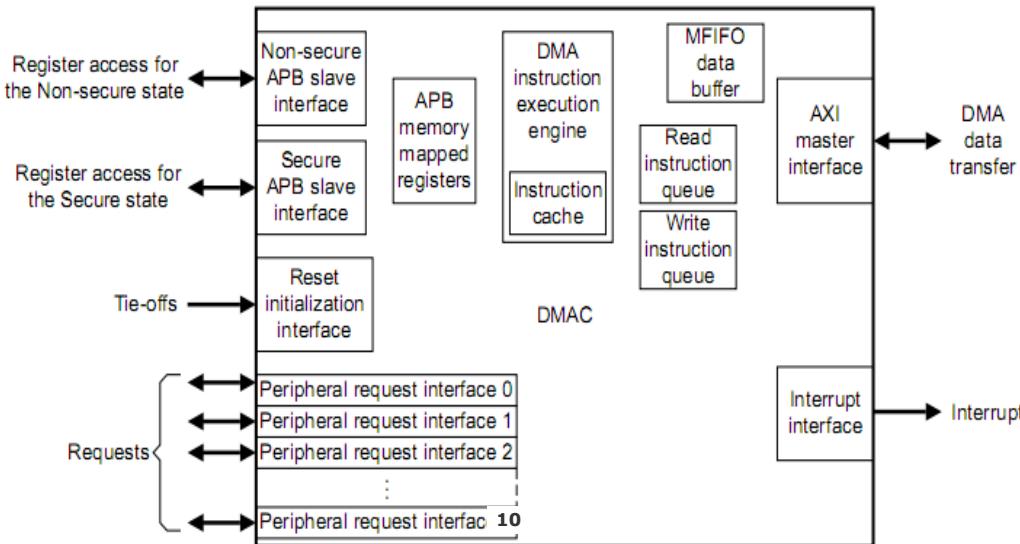


Fig. 10-1 Block diagram of DMAC\_BUS

As the DMAC\_BUS supports Trustzone technology, so dual APB interfaces enable the operation of the DMAC\_BUS to be partitioned into the secure state and Non-secure state. You can use the APB interfaces to access status registers and also directly execute instructions in the DMAC\_BUS. The default interface after reset is secure apb interface.

## 10.3 Function Description

### 10.3.1 Introduction

The DMAC contains an instruction processing block that enables it to process program code that controls a DMA transfer. The program code is stored in a region of system memory that the DMAC accesses using its AXI interface. The DMAC stores instructions temporarily in a cache.

DMAC\_BUS supports 6 channels, each channel capable of supporting a single concurrent thread of DMA operation. In addition, a single DMA manager thread exists, and you can use it to initialize the DMA channel threads. The DMAC executes up to one instruction for each AXI clock cycle. To ensure that it regularly executes each active thread, it alternates by processing the DMA manager thread and then a DMA channel thread. It uses a round-robin process when selecting the next active DMA channel thread to execute.

The DMAC uses variable-length instructions that consist of one to six bytes. It provides a separate Program Counter (PC) register for each DMA channel. When a thread requests an instruction from an address, the cache performs a look-up. If a cache hit occurs, then the cache immediately provides the data. Otherwise, the thread is stalled while the DMAC uses the AXI interface to perform a cache line fill. If an instruction is greater than 4 bytes, or spans the end of a cache line, the DMAC performs multiple cache accesses to fetch the instruction.

When a cache line fill is in progress, the DMAC enables other threads to access the cache, but if another cache miss occurs, this stalls the pipeline until the first line fill is complete.

When a DMA channel thread executes a load or store instruction, the DMAC adds the instruction to the relevant read or write queue. The DMAC uses these queues as an instruction storage buffer prior to it issuing the instructions on the AXI bus. The DMAC also contains a Multi First-In-First-Out (MFIFO) data buffer that it uses to store data that it reads, or writes, during a DMA transfer.

### **10.3.2 Operating states**

Figure shows the operating states for the DMA manager thread and DMA channel threads.

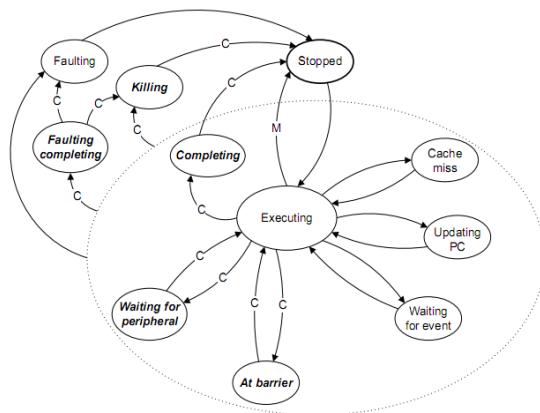


Fig. 10-2 DMAC\_BUS operation states

*Note:*

*arcs with no letter designator indicate state transitions for the DMA manager and DMA channel threads, otherwise use is restricted as follows:*

### C DMA channel threads only.

*M DMA manager thread only.*

After the DMAC exits from reset, it sets all DMA channel threads to the stopped state, and the status of boot\_from\_pc(tie-off interface of dmac) controls the DMA manager thread state:

boot\_from\_pc is LOW :DMA manager thread moves to the Stopped state.  
boot from pc is HIGH :DMA manager thread moves to the Executing state.

## 10.4 Register Description

### **10.4.1 Register summary**

<b>Name</b>	<b>Offset</b>	<b>Size</b>	<b>Reset Value</b>	<b>Description</b>
DMAC_BUS_DSR	0x0000	W	0x0	DMA Status Register.
DMAC_BUS_DPC	0x0004	W	0x0	DMA Program Counter Register.
-	-	-	-	reserved
DMAC_BUS_INTE_N	0x0020	W	0x0	Interrupt Enable Register
DMAC_BUS_EVE_NT_RIS	0x0024	W	0x0	Event Status Register.
DMAC_BUS_INT_MIS	0x0028	W	0x0	Interrupt Status Register
DMAC_BUS_INTC_LR	0x002C	W	0x0	Interrupt Clear Register
DMAC_BUS_FSR_D	0x0030	W	0x0	Fault Status DMA Manager Register.
DMAC_BUS_FSR_C	0x0034	W	0x0	Fault Status DMA Channel Register.
DMAC_BUS_FTR_D	0x0038	W	0x0	Fault Type DMA Manager Register.
-	-	-	-	reserved
DMAC_BUS_FTR0	0x0040	W	0x0	Fault type for DMA Channel 0

DMAC_BUS_FTR1	0x0044	W	0x0	Fault type for DMA Channel 1
DMAC_BUS_FTR2	0x0048	W	0x0	Fault type for DMA Channel 2
DMAC_BUS_FTR3	0x004C	W	0x0	Fault type for DMA Channel 3
DMAC_BUS_FTR4	0x0050	W	0x0	Fault type for DMA Channel 4
DMAC_BUS_FTR5	0x0054	W	0x0	Fault type for DMA Channel 5
-	-	-	-	reserved
DMAC_BUS_CSR0	0x0100	W	0x0	Channel Status for DMA Channel 0
DMAC_BUS_CSR1	0x0108	W	0x0	Channel Status for DMA Channel 1
DMAC_BUS_CSR2	0x0110	W	0x0	Channel Status for DMA Channel 2
DMAC_BUS_CSR3	0x0118	W	0x0	Channel Status for DMA Channel 3
DMAC_BUS_CSR4	0x0120	W	0x0	Channel Status for DMA Channel 4
DMAC_BUS_CSR5	0x0128	W	0x0	Channel Status for DMA Channel 5
DMAC_BUS_CPC0	0x0104	W	0x0	Channel PC for DMA Channel 0
DMAC_BUS_CPC1	0x010c	W	0x0	Channel PC for DMA Channel 1
DMAC_BUS_CPC2	0x0114	W	0x0	Channel PC for DMA Channel 2
DMAC_BUS_CPC3	0x011c	W	0x0	Channel PC for DMA Channel 3
DMAC_BUS_CPC4	0x0124	W	0x0	Channel PC for DMA Channel 4
DMAC_BUS_CPC5	0x012c	W	0x0	Channel PC for DMA Channel 5
DMAC_BUS_SAR0	0x0400	W	0x0	Source Address for DMA Channel 0
DMAC_BUS_SAR1	0x0420	W	0x0	Source Address for DMA Channel 1
DMAC_BUS_SAR2	0x0440	W	0x0	Source Address for DMA Channel 2
DMAC_BUS_SAR3	0x0460	W	0x0	Source Address for DMA Channel 3
DMAC_BUS_SAR4	0x0480	W	0x0	Source Address for DMA Channel 4
DMAC_BUS_SAR5	0x04a0	W	0x0	Source Address for DMA Channel 5
DMAC_BUS_DAR0	0x0404	W	0x0	Dest Address for DMAChannel 0
DMAC_BUS_DAR1	0x0424	W	0x0	Dest Address for DMAChannel 1
DMAC_BUS_DAR2	0x0444	W	0x0	Dest Address for DMAChannel 2
DMAC_BUS_DAR3	0x0464	W	0x0	Dest Address for DMAChannel 3
DMAC_BUS_DAR4	0x0484	W	0x0	Dest Address for DMAChannel 4
DMAC_BUS_DAR5	0x04a4	W	0x0	Dest Address for DMAChannel 5
DMAC_BUS_CCR0	0x0408	W	0x0	Channel Control for DMA Channel 0

DMAC_BUS_CCR_1	0x0428	W	0x0	Channel Control for DMA Channel 1
DMAC_BUS_CCR_2	0x0448	W	0x0	Channel Control for DMA Channel 2
DMAC_BUS_CCR_3	0x0468	W	0x0	Channel Control for DMA Channel 3
DMAC_BUS_CCR_4	0x0488	W	0x0	Channel Control for DMA Channel 4
DMAC_BUS_CCR_5	0x04a8	W	0x0	Channel Control for DMA Channel 5
DMAC_BUS_LC0_0	0x040C	W	0x0	Loop Counter 0 for DMA Channel 0
DMAC_BUS_LC0_1	0x042C	W	0x0	Loop Counter 0 for DMA Channel 1
DMAC_BUS_LC0_2	0x044C	W	0x0	Loop Counter 0 for DMA Channel 2
DMAC_BUS_LC0_3	0x046C	W	0x0	Loop Counter 0 for DMA Channel 3
DMAC_BUS_LC0_4	0x048C	W	0x0	Loop Counter 0 for DMA Channel 4
DMAC_BUS_LC0_5	0x04aC	W	0x0	Loop Counter 0 for DMA Channel 5
DMAC_BUS_LC1_0	0x0410	W	0x0	Loop Counter 1 for DMA Channel 0
DMAC_BUS_LC1_1	0x0430	W	0x0	Loop Counter 1 for DMA Channel 1
DMAC_BUS_LC1_2	0x0450	W	0x0	Loop Counter 1 for DMA Channel 2
DMAC_BUS_LC1_3	0x0470	W	0x0	Loop Counter 1 for DMA Channel 3
DMAC_BUS_LC1_4	0x0490	W	0x0	Loop Counter 1 for DMA Channel 4
DMAC_BUS_LC1_5	0x04b0	W	0x0	Loop Counter 1 for DMA Channel 5
-	-	-	-	reserved
DMAC_BUS_DBG_ST DMAC_BUS_ATU_S	0x0D00	W	0x0	Debug Status Register.
DMAC_BUS_DBG_CMD	0x0D04	W	0x0	Debug Command Register.
DMAC_BUS_DBG_INST0	0x0D08	W	0x0	Debug Instruction-0 Register.
DMAC_BUS_DBG_INST1	0x0D0C	W	0x0	Debug Instruction-1 Register.
DMAC_BUS_CR0	0x0E00	W		Configuration Register 0.
DMAC_BUS_CR1	0x0E04	W		Configuration Register 1.
DMAC_BUS_CR2	0x0E08	W		Configuration Register 2.
DMAC_BUS_CR3	0x0E0C	W		Configuration Register 3.
DMAC_BUS_CR4	0x0E10	W		Configuration Register 4.
DMAC_BUS_CRD_n	0x0E14	W		Configuration Register Dn.
DMAC_BUS_WD	0x0E80	W	0x0	Watchdog Register.

Notes:

Size: **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** -WORD (32 bits) access

## 10.4.2 Detail Register Description

### DMAC\_BUS\_DSR

Address: Operational Base+0x0

DMA Manager Status Register

Bit	Attr	Reset Value	Description
31:10	-	-	Reserved
9	R	0x0	Provides the security status of the DMA manager thread: 0 = DMA manager operates in the Secure state 1 = DMA manager operates in the Non-secure state.
8:4	R	0x0	When the DMA manager thread executes a DMAWFE instruction, it waits for the following event to occur: b00000 = event[0] b00001 = event[1] b00010 = event[2] ... b11111 = event[31].
3:0	R	0x0	The operating state of the DMA manager: b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = Waiting for event b0101-b1110 = reserved b1111 = Faulting.

### DMAC\_BUS\_DPC

Address: Operational Base+0x4

DMA Program Counter Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Program counter for the DMA manager thread

### DMAC\_BUS\_INTEN

Address: Operational Base+0x20

Interrupt Enable Register

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Program the appropriate bit to control how the DMAC responds when it executes DMASEV: Bit [N] = 0 If the DMAC executes DMASEV for the event-interrupt resource N then the DMAC signals event N to all of the threads. Set bit [N] to 0 if your system design does not use irq[N] to signal an interrupt request.  Bit [N] = 1 If the DMAC executes DMASEV for the event-interrupt resource N then the DMAC sets irq[N] HIGH. Set bit [N] to 1 if your system designer requires irq[N] to signal an interrupt request.

### DMAC\_BUS\_EVENT\_RIS

Address: Operational Base+0x24

Event-Interrupt Raw Status Register

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:0	R	0x0	Returns the status of the event-interrupt resources: Bit [N] = 0 Event N is inactive or irq[N] is LOW. Bit [N] = 1 Event N is active or irq[N] is HIGH.
------	---	-----	---

**DMAC\_BUS\_INTMIS**

Address: Operational Base+0x28

Interrupt Status Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Provides the status of the interrupts that are active in the DMAC: Bit [N] = 0 Interrupt N is inactive and therefore irq[N] is LOW. Bit [N] = 1 Interrupt N is active and therefore irq[N] is HIGH

**DMAC\_BUS\_INTCLR**

Address: Operational Base+0x2c

Interrupt Clear Register

Bit	Attr	Reset Value	Description
31:0	W	0x0	Controls the clearing of the irq outputs: Bit [N] = 0 The status of irq[N] does not change. Bit [N] = 1 The DMAC sets irq[N] LOW if the INTEN Register programs the DMAC to signal an interrupt. Otherwise, the status of irq[N] does not change.

**DMAC\_BUS\_FSRD**

Address: Operational Base+0x30

Fault Status DMA Manager Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Provides the fault status of the DMA manager. Read as: 0 = the DMA manager thread is not in the Faulting state 1 = the DMA manager thread is in the Faulting state.

**DMAC\_BUS\_FSRC**

Address: Operational Base+0x34

Fault Status DMA Channel Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Each bit provides the fault status of the corresponding channel. Read as: Bit [N] = 0 No fault is present on DMA channel N. Bit [N] = 1 DMA channel N is in the Faulting or Faulting completing state.

**DMAC\_BUS\_FTRD**

Address: Operational Base+0x38

Fault Type DMA Manager Register

Bit	Attr	Reset Value	Description
31	-	-	reserved
30	R	0x0	If the DMA manager aborts, this bit indicates if the erroneous instruction was read from the

			system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface.
29:17	-	-	reserved
16	R	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA manager performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response
15:6	-	-	reserved
5	R	0x0	Indicates if the DMA manager was attempting to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = DMA manager has appropriate security to execute DMAWFE or DMASEV 1 = a DMA manager thread in the Non-secure state attempted to execute either: <ul style="list-style-type: none"><li>• DMAWFE to wait for a secure event</li><li>• DMASEV to create a secure event or secure interrupt</li></ul>
4	R	0x0	Indicates if the DMA manager was attempting to execute DMAGO with inappropriate security permissions: 0 = DMA manager has appropriate security to execute DMAGO 1 = a DMA manager thread in the Non-secure state attempted to execute DMAGO to create a DMA channel operating in the Secure state.
3:2	-	-	reserved
1	R	0x0	Indicates if the DMA manager was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand.
0	R	0x0	Indicates if the DMA manager was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction.

**DMAC\_BUS\_FTR0~DMAC\_BUS\_FTR5**

Address: Operational Base+0x40

Operational Base+0x44

Operational Base+0x48

Operational Base+0x4c

Operational Base+0x50

Operational Base+0x54

Fault Type DMA Channel Register

Bit	Attr	Reset Value	Description
31	R	0x0	Indicates if the DMA channel has locked-up because of resource starvation: 0 = DMA channel has adequate resources 1 = DMA channel has locked-up because of insufficient resources. This fault is an imprecise abort

30	R	0x0	If the DMA channel aborts, this bit indicates if the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface. This fault is an imprecise abort but the bit is only valid when a precise abort occurs.
29:19	-	-	reserved
18	R	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs a data read: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort
17	R	0x0	Indicates the AXI response that the DMAC receives on the BRESP bus, after the DMA channel thread performs a data write: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
16	R	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is a precise abort.
15:14	-	-	reserved
13	R	0x0	Indicates if the MFIFO did not contain the data to enable the DMAC to perform the DMAST: 0 = MFIFO contains all the data to enable the DMAST to complete 1 = previous DMA LDs have not put enough data in the MFIFO to enable the DMAST to complete. This fault is a precise abort.
12	R	0x0	Indicates if the MFIFO prevented the DMA channel thread from executing DMA LD or DMA ST. Depending on the instruction: DMA LD 0 = MFIFO contains sufficient space 1 = MFIFO is too small to hold the data that DMA LD requires. DMA ST 0 = MFIFO contains sufficient data 1 = MFIFO is too small to store the data to enable DMA ST to complete. This fault is an imprecise abort
11:8	-	-	reserved
7	R	0x0	Indicates if a DMA channel thread, in the Non-secure state, attempts to program the CCRn Register to perform a secure read or secure write: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to perform a secure read or secure write. This fault is a precise abort

6	R	0x0	Indicates if a DMA channel thread, in the Non-secure state, attempts to execute DMAWFP, DMALDP, DMASTP, or DMAFLUSHP with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: <ul style="list-style-type: none"><li>• DMAWFP to wait for a secure peripheral</li><li>• DMALDP or DMASTP to notify a secure peripheral</li><li>• DMAFLUSHP to flush a secure peripheral.</li></ul> This fault is a precise abort.
5	R	0x0	Indicates if the DMA channel thread attempts to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: <ul style="list-style-type: none"><li>• DMAWFE to wait for a secure event</li><li>• DMASEV to create a secure event or secure interrupt.</li></ul> This fault is a precise abort.
4:2	-	-	reserved
1	R	0x0	Indicates if the DMA channel thread was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand. This fault is a precise abort.
0	R	0x0	Indicates if the DMA channel thread was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction. This fault is a precise abort

**DMAC\_BUS\_CSR0~DMAC\_BUS\_CSR5**

Address: Operational Base+0x100

Operational Base+0x108

Operational Base+0x110

Operational Base+0x118

Operational Base+0x120

Operational Base+0x128

## Channel Status Registers

Bit	Attr	Reset Value	Description
31:22	-	-	reserved
21	R	0x0	The channel non-secure bit provides the security of the DMA channel: 0 = DMA channel operates in the Secure state 1 = DMA channel operates in the Non-secure state
20:16	-	-	reserved
15	R	0x0	When the DMA channel thread executes DMAWFP this bit indicates if the periph operand was set:

			0 = DMAWFP executed with the periph operand not set 1 = DMAWFP executed with the periph operand set
14	R	0x0	When the DMA channel thread executes DMAWFP this bit indicates if the burst or single operand were set: 0 = DMAWFP executed with the single operand set 1 = DMAWFP executed with the burst operand set.
13:9	-	-	reserved
8:4	R	0x0	If the DMA channel is in the Waiting for event state or the Waiting for peripheral state then these bits indicate the event or peripheral number that the channel is waiting for: b00000 = DMA channel is waiting for event, or peripheral, 0 b00001 = DMA channel is waiting for event, or peripheral, 1 b00010 = DMA channel is waiting for event, or peripheral, 2 . . . b11111 = DMA channel is waiting for event, or peripheral, 31
3:0	R	0x0	The channel status encoding is: b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = Waiting for event b0101 = At barrier b0110 = reserved b0111 = Waiting for peripheral b1000 = Killing b1001 = Completing b1010-b1101 = reserved b1110 = Faulting completing b1111 = Faulting

**DMAC\_BUS\_CPC0~DMAC\_BUS\_CPC5**

Address: Operational Base+0x104

Operational Base+0x10c

Operational Base+0x114

Operational Base+0x11c

Operational Base+0x124

Operational Base+0x12c

## Channel Program Counter Registers

Bit	Attr	Reset Value	Description
31:0	R	0x0	Program counter for the DMA channel n thread

**DMAC\_BUS\_SAR0~DMAC\_BUS\_SAR5**

Address: Operational Base+0x400

Operational Base+0x420

Operational Base+0x440  
 Operational Base+0x460  
 Operational Base+0x480  
 Operational Base+0x4a0

## Source Address Registers

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x0	Address of the source data for DMA channel n

**DMAC\_BUS\_DAR0~DMAC\_BUS\_DAR5**

Address: Operational Base+0x404  
 Operational Base+0x424  
 Operational Base+0x444  
 Operational Base+0x464  
 Operational Base+0x484  
 Operational Base+0x4a4

## DestinationAddress Registers

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x0	Address of the Destinationdata for DMA channel n

**DMAC\_BUS\_CCR0~DMAC\_BUS\_CCR5**

Address: Operational Base+0x408  
 Operational Base+0x428  
 Operational Base+0x448  
 Operational Base+0x468  
 Operational Base+0x488  
 Operational Base+0x4a8

## Channel Control Registers

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	-	-	reserved
27:25	R	0x0	Programs the state of AWCACHE[3,1:0]a when the DMAC writes the destination data. Bit [27] 0 = AWCACHE[3] is LOW 1 = AWCACHE[3] is HIGH. Bit [26] 0 = AWCACHE[1] is LOW 1 = AWCACHE[1] is HIGH. Bit [25] 0 = AWCACHE[0] is LOW 1 = AWCACHE[0] is HIGH
24:22	R	0x0	Programs the state of AWPROT[2:0]a when the DMAC writes the destination data. Bit [24] 0 = AWPROT[2] is LOW 1 = AWPROT[2] is HIGH. Bit [23] 0 = AWPROT[1] is LOW 1 = AWPROT[1] is HIGH. Bit [22] 0 = AWPROT[0] is LOW 1 = AWPROT[0] is HIGH
21:18	R	0x0	For each burst, these bits program the number of data transfers that the DMAC performs when it writes the destination data: b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers. The total number of bytes that the DMAC writes

			out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size
17:15	R	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC writes to the destination:</p> <ul style="list-style-type: none"> <li>b000 = writes 1 byte per beat</li> <li>b001 = writes 2 bytes per beat</li> <li>b010 = writes 4 bytes per beat</li> <li>b011 = writes 8 bytes per beat</li> <li>b100 = writes 16 bytes per beat</li> <li>b101-b111 = reserved.</li> </ul> <p>The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size.</p>
14	R	0x0	<p>Programs the burst type that the DMAC performs when it writes the destination data:</p> <ul style="list-style-type: none"> <li>0 = Fixed-address burst. The DMAC signals AWBURST[0] LOW.</li> <li>1 = Incrementing-address burst. The DMAC signals AWBURST[0] HIGH.</li> </ul>
13:11	R	0x0	<p>Set the bits to control the state of ARCACHE[2:0]a when the DMAC reads the source data.</p> <ul style="list-style-type: none"> <li>Bit [13] 0 = ARCACHE[2] is LOW 1 = ARCACHE[2] is HIGH.</li> <li>Bit [12] 0 = ARCACHE[1] is LOW 1 = ARCACHE[1] is HIGH.</li> <li>Bit [11] 0 = ARCACHE[0] is LOW 1 = ARCACHE[0] is HIGH.</li> </ul>
10:8	R	0x0	<p>Programs the state of ARPROT[2:0]a when the DMAC reads the source data.</p> <ul style="list-style-type: none"> <li>Bit [10] 0 = ARPROT[2] is LOW 1 = ARPROT[2] is HIGH.</li> <li>Bit [9] 0 = ARPROT[1] is LOW 1 = ARPROT[1] is HIGH.</li> <li>Bit [8] 0 = ARPROT[0] is LOW 1 = ARPROT[0] is HIGH.</li> </ul>
7:4	R	0x0	<p>For each burst, these bits program the number of data transfers that the DMAC performs when it reads the source data:</p> <ul style="list-style-type: none"> <li>b0000 = 1 data transfer</li> <li>b0001 = 2 data transfers</li> <li>b0010 = 3 data transfers</li> <li>.</li> <li>.</li> <li>.</li> <li>b1111 = 16 data transfers.</li> </ul> <p>The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size</p>
3:1	R	0x0	<p>For each beat within a burst, it programs the number of bytes that the DMAC reads from the source:</p> <ul style="list-style-type: none"> <li>b000 = reads 1 byte per beat</li> </ul>

			b001 = reads 2 bytes per beat b010 = reads 4 bytes per beat b011 = reads 8 bytes per beat b100 = reads 16 bytes per beat b101-b111 = reserved. The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size
0	R	0x0	Programs the burst type that the DMAC performs when it reads the source data: 0 = Fixed-address burst. The DMAC signals ARBURST[0] LOW. 1 = Incrementing-address burst. The DMAC signals ARBURST[0] HIGH

**DMAC\_BUS\_LC0\_0~DMAC\_BUS\_LC0\_5**

Address: Operational Base+0x40c

Operational Base+0x42c

Operational Base+0x44c

Operational Base+0x46c

Operational Base+0x48c

Operational Base+0x4ac

Loop Counter 0 Registers

Bit	Attr	Reset Value	Description
31:8	-	-	reserved
7:0	R	0x0	Loop counter 0 iterations

**DMAC\_BUS\_LC1\_0~DMAC\_BUS\_LC1\_5**

Address: Operational Base+0x410

Operational Base+0x430

Operational Base+0x450

Operational Base+0x470

Operational Base+0x490

Operational Base+0x4b0

Loop Counter 1 Registers

Bit	Attr	Reset Value	Description
31:8	-	-	reserved
7:0	R	0x0	Loop counter 1 iterations

**DMAC\_BUS\_DBGSTATUS**

Address: Operational Base+0xd00

Debug Status Register

Bit	Attr	Reset Value	Description
31:2	-	-	reserved
1:0	R	0x0	The debug encoding is as follows: b00 = execute the instruction that the DBGINST [1:0] Registers contain b01 = reserved b10 = reserved b11 = reserved.

**DMAC\_BUS\_DBGCMD**

Address: Operational Base+0xd04

Debug Command Register

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:2	-	-	reserved
1:0	W	0x0	The debug encoding is as follows: b00 = execute the instruction that the DBGINST [1:0] Registers contain b01 = reserved b10 = reserved b11 = reserved

**DMAC\_BUS\_DBGINST0**

Address: Operational Base+0xd08

Debug Instruction-0 Register

Bit	Attr	Reset Value	Description
31:24	W	0x0	Instruction byte 1
23:16	W	0x0	Instruction byte 0
15:11	-	-	reserved
10:8	W	0x0	DMA channel number: b000 = DMA channel 0 b001 = DMA channel 1 b010 = DMA channel 2 ... b111 = DMA channel 7
7:1	-	-	reserved
0	W	0x0	The debug thread encoding is as follows: 0 = DMA manager thread 1 = DMA channel.

**DMAC\_BUS\_DBGINST1**

Address: Operational Base+0xd0c

Debug Instruction-1 Register

Bit	Attr	Reset Value	Description
31:24	W	0x0	Instruction byte 5
23:16	W	0x0	Instruction byte 4
15:8	W	0x0	Instruction byte 3
7:0	W	0x0	Instruction byte 2

**DMAC\_BUS\_CRO**

Address: Operational Base+0xe00

Configuration Register 0

Bit	Attr	Reset Value	Description
31:22	-	-	reserved
21:17	R	0x2	Number of interrupt outputs that the DMAC provides: b00000 = 1 interrupt output, irq[0] b00001 = 2 interrupt outputs, irq[1:0] b00010 = 3 interrupt outputs, irq[2:0] . . . b11111 = 32 interrupt outputs, irq[31:0].
16:12	R	0x7	Number of peripheral request interfaces that the DMAC provides: b00000 = 1 peripheral request interface b00001 = 2 peripheral request interfaces b00010 = 3 peripheral request interfaces . . .

			b11111 = 32 peripheral request interfaces.
11:7	-	-	reserved
6:4	R	0x5	<p>Number of DMA channels that the DMAC supports:</p> <p>b000 = 1 DMA channel      b001 = 2 DMA channels      b010 = 3 DMA channels      .      .      .      b111 = 8 DMA channels.</p>
3	-	-	reserved
2	R	0x0	Indicates the status of the boot_manager_ns signal when the DMAC exited from reset: 0 = boot_manager_ns was LOW 1 = boot_manager_ns was HIGH.
1	R	0x0	Indicates the status of the boot_from_pc signal when the DMAC exited from reset: 0 = boot_from_pc was LOW 1 = boot_from_pc was HIGH
0	R	0x1	Supports peripheral requests: 0 = the DMAC does not provide a peripheral request interface 1 = the DMAC provides the number of peripheral request interfaces that the num_periph_req field specifies.

**DMAC\_BUS\_CR1**

Address: Operational Base+0xe04

Configuration Register 1

Bit	Attr	Reset Value	Description
31:8	-	-	reserved
7:4	R	0x5	[7:4] num_i-cache_lines Number of i-cache lines: b0000 = 1 i-cache line b0001 = 2 i-cache lines b0010 = 3 i-cache lines ... b1111 = 16 i-cache lines.
3	-	-	reserved
2:0	R	0x7	The length of an i-cache line: b000-b001 = reserved b010 = 4 bytes b011 = 8 bytes b100 = 16 bytes b101 = 32 bytes b110-b111 = reserved

**DMAC\_BUS\_CR2**

Address: Operational Base+0xe08

Configuration Register 2

Bit	Attr	Reset Value	Description
31:0	R	0x0	Provides the value of boot_addr[31:0] when the DMAC exited from reset

**DMAC\_BUS\_CR3**

Address: Operational Base+0xe0c

## Configuration Register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x0	<p>Provides the security state of an event-interrupt resource:</p> <p>Bit [N] = 0 Assigns event&lt;N&gt; or irq[N] to the Secure state.</p> <p>Bit [N] = 1 Assigns event&lt;N&gt; or irq[N] to the Non-secure state.</p>

**DMAC\_BUS\_CR4**

Address: Operational Base+0xe10

## Configuration Register 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x6	<p>Provides the security state of the peripheral request interfaces:</p> <p>Bit [N] = 0 Assigns peripheral request interface N to the Secure state.</p> <p>Bit [N] = 1 Assigns peripheral request interface N to the Non-secure state</p>

**DMAC\_BUS\_CRDn**

Address: Operational Base+0xe14

## DMA Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	-	-	reserved
29:20	R	0x20	<p>The number of lines that the data buffer contains:</p> <p>b000000000 = 1 line      b000000001 = 2 lines      ...      b111111111 = 1024 lines</p>
19:16	R	0x9	<p>The depth of the read queue:</p> <p>b0000 = 1 line      b0001 = 2 lines      .      .      b1111 = 16 lines.</p>
15	-	-	reserved
14:12	R	0x4	<p>Read issuing capability that programs the number of outstanding read transactions:</p> <p>b000 = 1      b001 = 2      ...      b111 = 8</p>
11:8	R	0x7	<p>The depth of the write queue:</p> <p>b0000 = 1 line      b0001 = 2 lines      ...      b1111 = 16 lines.</p>
7	-	-	reserved
6:4	R	0x3	<p>Write issuing capability that programs the number of outstanding write transactions:</p> <p>b000 = 1      b001 = 2      ...      b111 = 8</p>

3	-	-	reserved
2:0		0x3	The data bus width of the AXI interface: b000 = reserved b001 = reserved b010 = 32-bit b011 = 64-bit b100 = 128-bit b101-b111 = reserved.

**DMAC\_BUS\_WD**

Address: Operational Base+0xe80

DMA Watchdog Register

Bit	Attr	Reset Value	Description
-	-	-	reserved
0	RW	0x0	Controls how the DMAC responds when it detects a lock-up condition: 0 = the DMAC aborts all of the contributing DMA channels and sets irq_abort HIGH 1 = the DMAC sets irq_abort HIGH.

**10.5 Timing Diagram**

Following picture shows the relationship between dma\_req and dma\_ack.

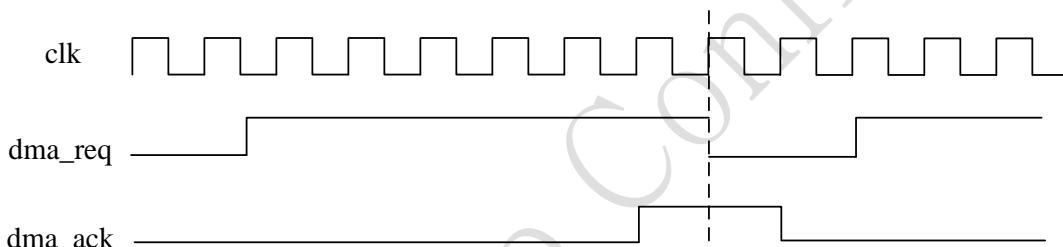


Fig. 10-3 DMAC\_BUS request and acknowledge timing

**10.6 Interface Description**

DMAC\_BUS has the following tie-off signals. It can be configured by GRF register or TZPC register. (Please refer to these two chapters to find how to configure)

interface	Reset value	Control source
boot_addr	0x0	Secure GRF
boot_from_pc	0x0	Secure GRF
boot_manager_ns	0x0	Secure GRF / TZPC
boot_irq_ns	0x0	Secure GRF / TZPC
boot_periph_ns	0x0	TZPC

**boot\_addr**

Configures the address location that contains the first instruction the DMAC executes, when it exits from reset.

**boot\_from\_pc**

Controls the location in which the DMAC executes its initial instruction, after it exits from reset:

- 0 = DMAC waits for an instruction from either APB interface  
1 = DMA manager thread executes the instruction that is located at the address that

**boot\_manager\_ns**

When the DMAC exits from reset, this signal controls the security state of the DMA manager thread:

- 0 = assigns DMA manager to the Secure state  
1 = assigns DMA manager to the Non-secure state.

**boot\_irq\_ns**

Controls the security state of an event-interrupt resource, when the DMAC exits from reset:

- boot\_irq\_ns[x] is LOW  
The DMAC assigns event<x> or irq[x] to the Secure state.  
boot\_irq\_ns[x] is HIGH  
The DMAC assigns event<x> or irq[x] to the Non-secure state.

**boot\_periph\_ns**

Controls the security state of a peripheral request interface, when the DMAC exits from reset:

- boot\_periph\_ns[x] is LOW  
The DMAC assigns peripheral request interface x to the Secure state.  
boot\_periph\_ns[x] is HIGH  
The DMAC assigns peripheral request interface x to the Non-secure state.

## 10.7 Application Notes

### 10.7.1 Using the APB slave interfaces

You must ensure that you use the appropriate APB interface, depending on the security state in which the boot\_manager\_ns initializes the DMAC to operate. For example, if the DMAC is in the secure state, you must issue the instruction using the secure APB interface, otherwise the DMAC ignores the instruction. You can use the secure APB interface, or the non-secure APB interface, to start or restart a DMA channel when the DMAC is in the Non-secure state.

The necessary steps to start a DMA channel thread using the debug instruction registers as following:

1. Create a program for the DMA channel.
2. Store the program in a region of system memory.
3. Poll the DBGSTATUS Register to ensure that debug is idle, that is, the dbgstatus bit is 0.
4. Write to the DBGINST0 Register and enter the:
  - Instruction byte 0 encoding for DMAGO.
  - Instruction byte 1 encoding for DMAGO.
  - Debug thread bit to 0. This selects the DMA manager thread.
5. Write to the DBGINST1 Register with the DMAGO instruction byte [5:2] data, see Debug Instruction-1 Register o. You must set these four bytes to the address of the first instruction in the program, that was written to system memory in step 2.
6. Writing zero to the DBGCMD Register. The DMAC starts the DMA channel thread and sets the dbgstatus bit to 1.

## 10.7.2 Security usage

When the DMAC exits from reset, the status of the configuration signals that tie-off signals which described in chapter 10.6.

### DMA manager thread is in the secure state

If the DNS bit is 0, the DMA manager thread operates in the secure state and it only performs secure instruction fetches. When a DMA manager thread in the secure state processes:

#### DMAGO

It uses the status of the ns bit, to set the security state of the DMA channel thread by writing to the CNS bit for that channel.

#### DMAWFE

It halts execution of the thread until the event occurs. When the event occurs, the DMAC continues execution of the thread, irrespective of the security state of the corresponding INS bit.

#### DMASEV

It sets the corresponding bit in the INT\_EVENT\_RIS Register, irrespective of the security state of the corresponding INS bit.

### DMA manager thread is in the Non-secure state

If the DNS bit is 1, the DMA manager thread operates in the Non-secure state, and it only performs non-secure instruction fetches. When a DMA manager thread in the Non-secure state processes:

#### DMAGO

The DMAC uses the status of the ns bit, to control if it starts a DMA channel thread. If:

ns = 0

The DMAC does not start a DMA channel thread and instead it:

1. Executes a NOP.
2. Sets the FSRD Register, see Fault Status DMA Manager
3. Sets the dmago\_err bit in the FTRD Register, see Fault Type DMA Manager Register.
4. Moves the DMA manager to the Faulting state.

ns = 1

The DMAC starts a DMA channel thread in the Non-secure state and programs the CNS bit to be non-secure.

#### DMAWFE

The DMAC uses the status of the corresponding INS bit, in the CR3 Register, to control if it waits for the event. If:

INS = 0

The event is in the Secure state. The DMAC:

1. Executes a NOP.
2. Sets the FSRD Register, see Fault Status DMA Manager Register.
3. Sets the mgr\_evnt\_err bit in the FTRD Register, see Fault Type DMA Manager Register.
4. Moves the DMA manager to the Faulting state.

INS = 1

The event is in the Non-secure state. The DMAC halts execution of the thread and waits for the event to occur.

### **DMASEV**

The DMAC uses the status of the corresponding INS bit, in the CR3Register, to control if it creates the event-interrupt. If:

INS = 0

The event-interrupt resource is in the Secure state. The DMAC:

1. Executes a NOP.
2. Sets the FSRD Register, see Fault Status DMA Manager Register.
3. Sets the mgr\_evnt\_err bit in the FTRD Register, see Fault Type DMA Manager Register.
4. Moves the DMA manager to the Faulting state.

INS = 1

The event-interrupt resource is in the Non-secure state. The DMAC creates the event-interrupt.

### **DMA channel thread is in the secure state**

When the CNS bit is 0, the DMA channel thread is programmed to operate in the Secure state and it only performs secure instruction fetches.

When a DMA channel thread in the secure state processes the following instructions:

### **DMAWFE**

The DMAC halts execution of the thread until the event occurs. When the event occurs, the DMAC continues execution of the thread, irrespective of the security state of the corresponding INS bit, in the CR3 Register.

### **DMASEV**

The DMAC creates the event-interrupt, irrespective of the security state of the corresponding INS bit, in the CR3 Register.

### **DMAWFP**

The DMAC halts execution of the thread until the peripheral signals a DMA request. When this occurs, the DMAC continues execution of the thread, irrespective of the security state of the corresponding PNS bit, in the CR4 Register.

### **DMALDP, DMASTP**

The DMAC sends a message to the peripheral to communicate that data transfer is complete, irrespective of the security state of the corresponding PNS bit, in the CR4 Register.

### **DMAFLUSHP**

The DMAC clears the state of the peripheral and sends a message to the peripheral to resend its level status, irrespective of the security state of the corresponding PNS bit, in the CR4 Register.

When a DMA channel thread is in the Secure state, it enables the DMAC to perform secure and non-secure AXI accesses

### **DMA channel thread is in the Non-secure state**

When the CNS bit is 1, the DMA channel thread is programmed to operate in the Non-secure state and it only performs non-secure instruction fetches.

When a DMA channel thread in the Non-secure state processes the following instructions:

**DMAWFE**

The DMAC uses the status of the corresponding INS bit, in the CR3 Register, to control if it waits for the event. If:

INS = 0

The event is in the Secure state. The DMAC:

1. Executes a NOP.
2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number. See Fault Status DMA Channel Register.
3. Sets the ch\_evnt\_err bit in the FTRn Register, see Fault Type DMA Channel Registers.
4. Moves the DMA channel to the Faulting completing state.

INS = 1

The event is in the Non-secure state. The DMAC halts execution of the thread and waits for the event to occur.

**DMASEV**

The DMAC uses the status of the corresponding INS bit, in the CR3 Register, to control if it creates the event. If:

INS = 0

The event-interrupt resource is in the Secure state. The DMAC:

1. Executes a NOP.
2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number. See Fault Status DMA Channel Register.
3. Sets the ch\_evnt\_err bit in the FTRn Register, see Fault Type DMA Channel Registers .
4. Moves the DMA channel to the Faulting completing state.

INS = 1

The event-interrupt resource is in the Non-secure state. The DMAC creates the event-interrupt.

**DMAWFP**

The DMAC uses the status of the corresponding PNS bit, in the CR4 Register, to control if it waits for the peripheral to signal a request. If:

PNS = 0

The peripheral is in the Secure state. The DMAC:

1. Executes a NOP.
2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number. See Fault Status DMA Channel Register.
3. Sets the ch\_periph\_err bit in the FTRn Register, see Fault Type DMA Channel Registers.
4. Moves the DMA channel to the Faulting completing state.

PNS = 1

The peripheral is in the Non-secure state. The DMAC halts execution of the thread and waits for the peripheral to signal a request.

**DMALDP, DMASTP**

The DMAC uses the status of the corresponding PNS bit, in the CR4 Register, to control if it sends an acknowledgement to the peripheral. If:

PNS = 0

The peripheral is in the Secure state. The DMAC:

1. Executes a NOP.

2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number.  
See Fault Status DMA Channel Register.
3. Sets the ch\_periph\_err bit in the FTRn Register, see Fault Type DMA Channel Registers.
4. Moves the DMA channel to the Faulting completing state.

PNS = 1

The peripheral is in the Non-secure state. The DMAC sends a message to the peripheral to communicate when the data transfer is complete.

### **DMAFLUSHP**

The DMAC uses the status of the corresponding PNS bit, in the CR4 Register, to control if it sends a flush request to the peripheral. If:

PNS = 0

The peripheral is in the Secure state. The DMAC:

1. Executes a NOP.
2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number.  
See Fault Status DMA Channel Register.
3. Sets the ch\_periph\_err bit in the FTRn Register, see Fault Type DMA Channel Registers.
4. Moves the DMA channel to the Faulting completing state.

PNS = 1

The peripheral is in the Non-secure state. The DMAC clears the state of the peripheral and sends a message to the peripheral to resend its level status.

When a DMA channel thread is in the Non-secure state, and a DMAMOV CCR instruction attempts to program the channel to perform a secure AXI transaction, the DMAC:

1. Executes a DMANOP.
2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number.  
See Fault Status DMA Channel Register.
3. Sets the ch\_rdwr\_err bit in the FTRn Register, see Fault Type DMA Channel Registers.
4. Moves the DMA channel thread to the Faulting completing state.

### **10.7.3 Programming restrictions**

#### **Fixed unaligned bursts**

The DMAC does not support fixed unaligned bursts. If you program the following conditions, the DMAC treats this as a programming error:

Unaligned read

- src\_inc field is 0 in the CCRn Register
- the SARn Register contains an address that is not aligned to the size of data that the src\_burst\_size field contain

Unaligned write

- dst\_inc field is 0 in the CCRn Register
- the DARn Register contains an address that is not aligned to the size of data that the dst\_burst\_size field contains

#### **Endian swap size restrictions**

If you program the endian\_swap\_size field in the CCRn Register, to enable a DMA channel to perform an endian swap then you must set the corresponding SARn Register and the corresponding DARn Register to contain an address that is aligned to the value that the endian\_swap\_size field contains.

## Updating DMA channel control registers during a DMA cyclerestrictions

Prior to the DMAC executing a sequence of DMA LD and DMA ST instructions, the values you program in to the CCRn Register, SARn Register, and DARn Register control the data byte lane manipulation that the DMAC performs when it transfers the data from the source address to the destination address. You'd better not update these registers during a DMA cycle.

## Resource sharing between DMA channels

DMA channel programs share the MFIFO data storage resource. You must not start a set of concurrently running DMA channel programs with a resource requirement that exceeds the configured size of the MFIFO. If you exceed this limit then the DMAC might lock up and generate a Watchdog abort.

### 10.7.4 Unaligned transfers may be corrupted

For a configuration with more than one channel, if any of channels 1 to 7 is performing transfers between certain types of misaligned source and destination addresses, then the output data may be corrupted by the action of channel 0.

Data corruption might occur if all of the following are true:

1. Two beats of AXI read data are received for one of channels 1 to 7.
2. Source and destination address alignments mean that each read data beat is split across two lines in the data buffer (see Splitting data, below).
3. There is one idle cycle between the two read data beats .
4. Channel 0 performs an operation that updates channel control information during this idle cycle (see Updates to channel control information, below)

## Splitting data

Depending upon the programmed values for the DMA transfer, one beat of read data from the AXI interface need to be split across two lines in the internal data buffer. This occurs when the read data beat contains datbytes which will be written to addresses that wrap around at the AXI interface data width, so that these bytes could not be transferred by a single AXI write data beat of the full interface width.

Most applications of DMA-330 do not split data in this way, so are NOT vulnerable to data corruption from this defect.

The following cases are NOT vulnerable to data corruption because they do not split data:

- Byte lane offset between source and destination addresses is 0 When source and destination addresses have the same byte lane alignment, the offset is 0 and a wrap operation that splits data cannot occur.
- Byte lane offset between source and destination addresses is a multiple of source size

Source size in CCRn	Allowed offset between SARn and DARn
SS8	any offset allowed.
SS16	0,2,4,6,8,10,12,14
SS32	0,4,8,12
SS64	0,8

### 10.7.5 Interrupt shares between channel.

As the DMAC\_BUS does not record which channel (or list of channels) have asserted an interrupt. So it will depend on your program and whether any of the visible information for that program can be used to determine progress, and help identify the interrupt source.

There are 4 likely information sources that can be used to determine the progress made by a program:

- Program counter (PC)
- Source address
- Destination address
- Loop counters (LC)

For example, a program might emit an interrupt each time that it iterates around a loop. In this case, the interrupt service routine (ISR) would need to store the loop value of each channel when it is called, and then compare against the new value when it is next called. A change in value would indicate that the program has progressed.

The ISR must be carefully written to ensure that no interrupts are lost. The sequence of operations is as follows:

1. Disable interrupts
2. Immediately clear the interrupt in DMA-330
3. Check the relevant registers for both channels to determine which must be serviced
4. Take appropriate action for the channels
5. Re-enable interrupts and exit ISR

## 10.7.6 Instruction sets

Table 10-2 DMAC Instruction sets

Mnemonic	Instruction	Thread usage: • M = DMA manager • C = DMA channel
DMAADDH	Add Halfword	C
DMAEND	End	M/C
DMAFLUSHP	Flush and notify Peripheral	C
DMAGO	Go	M
DMAKILL	Kill	C
DMALD	Load	C
DMALDP	Load Peripheral	C
DMALP	Loop	C
DMALPEND	Loop End	C
DMALPFE	Loop Forever	C
DMAMOV	Move	C
DMANOP	No operation	M/C
DMARMB	Read Memory Barrier	C
DMASEV	Send Event	M/C
DMAST	Store	C
DMASTP	Store and notify Peripheral	C
DMASTZ	Store Zero	C
DMAWFE	Wait For Event M	M/C
DMAWFP	Wait For Peripheral	C
DMAWMB	Write Memory Barrier	C
DMAADNH	Add Negative Halfword	C

## 10.7.7 Assembler directives

In this document, only DMMADNH instruction is took as an example to show the way the instruction assembled. *For the other instructions , please refer to pl330\_trm.pdf.*

### DMAADNH

Add Negative Halfword adds an immediate negative 16-bit value to the SARn Register or D ARn Register, for the DMA channel thread. This enables the DMAC to support 2D DMA oper

ations, or reading or writing an area of memory in a different order to naturally incrementing addresses. See Source Address Registers and Destination Address Registers.

The immediate unsigned 16-bit value is one-extended to 32 bits, to create a value that is the two's complement representation of a negative number between -65536 and -1, before the DMAC adds it to the address using 32-bit addition. The DMAC discards the carry bit so that addresses wrap from 0xFFFFFFFF to 0x00000000. The net effect is to subtract between 65536 and 1 from the current value in the Source or Destination Address Register.

Following table shows the instruction encoding.

Imm[15:8]	Imm[7:0]	0	1	0	1	1	1	ra	0
-----------	----------	---	---	---	---	---	---	----	---

### **Assembler syntax**

DMAADNH <address\_register>, <16-bit immediate>

where:

<address\_register>

Selects the address register to use. It must be either:

SAR

SARn Register and sets ra to 0.

DAR

DARn Register and sets ra to 1.

<16-bit immediate>

The immediate value to be added to the <address\_register>.

You should specify the 16-bit immediate as the number that is to be represented in the instruction encoding. For example, DMAADNH DAR, 0xFFFF causes the value 0xFFFFFFF0 to be added to the current value of the Destination Address Register, effectively subtracting 16 from the DAR.

You can only use this instruction in a DMA channel thread.

### **10.7.8 MFIFO usage**

*For MFIFO usage , please refer to pl330\_trm.pdf*

# Chapter 11 DMA Controller for Peripheral System (DMAC\_PERI)

## 11.1 Overview

DMAC\_PERI does not support TrustZone technology and work under non-secure state only.

DMAC\_PERI is mainly used for data transfer of the following slaves: HSADC, UART BT, UART BB, UART GPS, UART EXP, SPI0, SPI1, SPI2.

Following table shows the DMAC\_PERI request mapping scheme.

Table 11-1 DMAC\_PERI Request Mapping Table

Req number	Source	Polarity
0	HSADC/TSI	High level
1	UART BT tx	High level
2	UART BT rx	High level
3	UART BB tx	High level
4	UART BB rx	High level
5	Reserved	
6	Reserved	
7	UART GPS tx	High level
8	UART GPS rx	High level
9	UART EXP tx	High level
10	UART EXP rx	High level
11	SPI0 tx	High level
12	SPI0 rx	High level
13	SPI1 tx	High level
14	SPI1 rx	High level
15	SPI2 tx	High level
16	SPI2 rx	High level
17	Reserved	
18	Reserved	
19	Reserved	

DMAC\_PERI supports the following features:

- Supports 20 peripheral request.
- Up to 64bits data size.
- 8 channel at the same time.
- Up to burst 16.
- 16 interrupts output and 1 abort output.
- Supports 64 MFIFO depth.

## 11.2 Block Diagram

Figure 11-1 shows the block diagram of DMAC\_PERI.

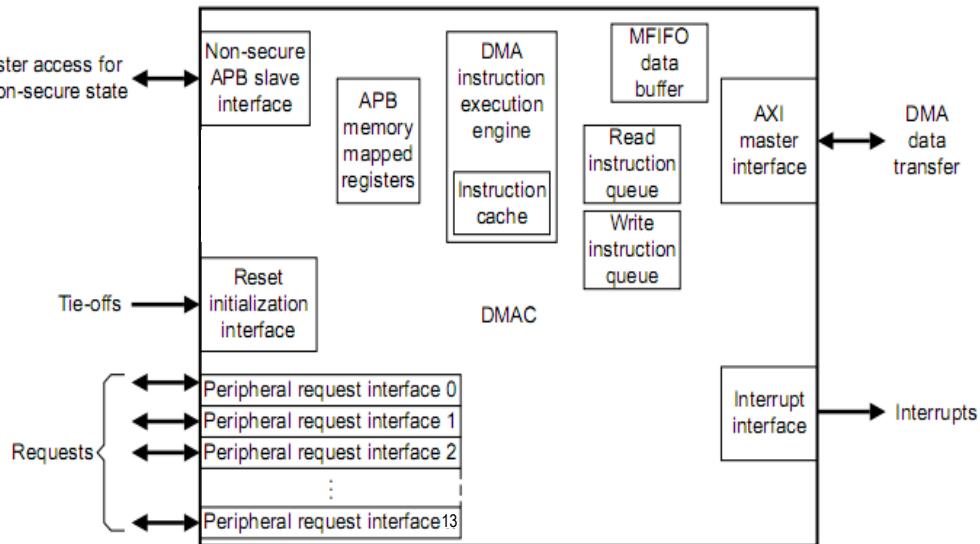


Fig. 11-1 Block diagram of DMAC\_PERI

## 11.3 Function Description

Please refer to chapter 10.3 for the similar description.

## 11.4 Register Description

### 11.4.1 Register summary

Name	Offset	Size	Reset Value	Description
DMAC_PERI_DSR	0x0000	W	0x0	DMA Status Register.
DMAC_PERI_DPC	0x0004	W	0x0	DMA Program Counter Register.
-	-	-	-	reserved
DMAC_PERI_INT_EN	0x0020	W	0x0	Interrupt Enable Register
DMAC_PERI_EVENT_RIS	0x0024	W	0x0	Event Status Register.
DMAC_PERI_INT_MIS	0x0028	W	0x0	Interrupt Status Register
DMAC_PERI_INT_CLR	0x002C	W	0x0	Interrupt Clear Register
DMAC_PERI_FSR_D	0x0030	W	0x0	Fault Status DMA Manager Register.
DMAC_PERI_FSR_C	0x0034	W	0x0	Fault Status DMA Channel Register.
DMAC_PERI_FTR_D	0x0038	W	0x0	Fault Type DMA Manager Register.
-	-	-	-	reserved
DMAC_PERI_FTR_0	0x0040	W	0x0	Fault type for DMA Channel 0
DMAC_PERI_FTR_1	0x0044	W	0x0	Fault type for DMA Channel 1
DMAC_PERI_FTR_2	0x0048	W	0x0	Fault type for DMA Channel 2
DMAC_PERI_FTR_3	0x004C	W	0x0	Fault type for DMA Channel 3

DMAC_PERI_FTR_4	0x0050	W	0x0	Fault type for DMA Channel 4
DMAC_PERI_FTR_5	0x0054	W	0x0	Fault type for DMA Channel 5
DMAC_PERI_FTR_6	0x0058	W	0x0	Fault type for DMA Channel 6
-	-	-	-	reserved
DMAC_PERI_CSR_0	0x0100	W	0x0	Channel Status for DMA Channel 0
DMAC_PERI_CSR_1	0x0108	W	0x0	Channel Status for DMA Channel 1
DMAC_PERI_CSR_2	0x0110	W	0x0	Channel Status for DMA Channel 2
DMAC_PERI_CSR_3	0x0118	W	0x0	Channel Status for DMA Channel 3
DMAC_PERI_CSR_4	0x0120	W	0x0	Channel Status for DMA Channel 4
DMAC_PERI_CSR_5	0x0128	W	0x0	Channel Status for DMA Channel 5
DMAC_PERI_CSR_6	0x0130	W	0x0	Channel Status for DMA Channel 6
DMAC_PERI_CPC_0	0x0104	W	0x0	Channel PC for DMA Channel 0
DMAC_PERI_CPC_1	0x010c	W	0x0	Channel PC for DMA Channel 1
DMAC_PERI_CPC_2	0x0114	W	0x0	Channel PC for DMA Channel 2
DMAC_PERI_CPC_3	0x011c	W	0x0	Channel PC for DMA Channel 3
DMAC_PERI_CPC_4	0x0124	W	0x0	Channel PC for DMA Channel 4
DMAC_PERI_CPC_5	0x012c	W	0x0	Channel PC for DMA Channel 5
DMAC_PERI_CPC_6	0x0134	W	0x0	Channel PC for DMA Channel 6
DMAC_PERI_SAR_0	0x0400	W	0x0	Source Address for DMA Channel 0
DMAC_PERI_SAR_1	0x0420	W	0x0	Source Address for DMA Channel 1
DMAC_PERI_SAR_2	0x0440	W	0x0	Source Address for DMA Channel 2
DMAC_PERI_SAR_3	0x0460	W	0x0	Source Address for DMA Channel 3
DMAC_PERI_SAR_4	0x0480	W	0x0	Source Address for DMA Channel 4
DMAC_PERI_SAR_5	0x04a0	W	0x0	Source Address for DMA Channel 5
DMAC_PERI_SAR_6	0x04c0	W	0x0	Source Address for DMA Channel 6
DMAC_PERI_DAR_0	0x0404	W	0x0	Dest Address for DMAChannel 0
DMAC_PERI_DAR_1	0x0424	W	0x0	Dest Address for DMAChannel 1
DMAC_PERI_DAR_2	0x0444	W	0x0	Dest Address for DMAChannel 2
DMAC_PERI_DAR	0x0464	W	0x0	Dest Address for DMAChannel

3				3
DMAC_PERI_DAR_4	0x0484	W	0x0	Dest Address for DMAChannel 4
DMAC_PERI_DAR_5	0x04a4	W	0x0	Dest Address for DMAChannel 5
DMAC_PERI_DAR_6	0x04c4	W	0x0	Dest Address for DMAChannel 6
DMAC_PERI_CCR_0	0x0408	W	0x0	Channel Control for DMA Channel 0
DMAC_PERI_CCR_1	0x0428	W	0x0	Channel Control for DMA Channel 1
DMAC_PERI_CCR_2	0x0448	W	0x0	Channel Control for DMA Channel 2
DMAC_PERI_CCR_3	0x0468	W	0x0	Channel Control for DMA Channel 3
DMAC_PERI_CCR_4	0x0488	W	0x0	Channel Control for DMA Channel 4
DMAC_PERI_CCR_5	0x04a8	W	0x0	Channel Control for DMA Channel 5
DMAC_PERI_CCR_6	0x04c8	W	0x0	Channel Control for DMA Channel 6
DMAC_PERI_LC0_0	0x040C	W	0x0	Loop Counter 0 for DMA Channel 0
DMAC_PERI_LC0_1	0x042C	W	0x0	Loop Counter 0 for DMA Channel 1
DMAC_PERI_LC0_2	0x044C	W	0x0	Loop Counter 0 for DMA Channel 2
DMAC_PERI_LC0_3	0x046C	W	0x0	Loop Counter 0 for DMA Channel 3
DMAC_PERI_LC0_4	0x048C	W	0x0	Loop Counter 0 for DMA Channel 4
DMAC_PERI_LC0_5	0x04aC	W	0x0	Loop Counter 0 for DMA Channel 5
DMAC_PERI_LC0_6	0x04cC	W	0x0	Loop Counter 0 for DMA Channel 6
DMAC_PERI_LC1_0	0x0410	W	0x0	Loop Counter 1 for DMA Channel 0
DMAC_PERI_LC1_1	0x0430	W	0x0	Loop Counter 1 for DMA Channel 1
DMAC_PERI_LC1_2	0x0450	W	0x0	Loop Counter 1 for DMA Channel 2
DMAC_PERI_LC1_3	0x0470	W	0x0	Loop Counter 1 for DMA Channel 3
DMAC_PERI_LC1_4	0x0490	W	0x0	Loop Counter 1 for DMA Channel 4
DMAC_PERI_LC1_5	0x04b0	W	0x0	Loop Counter 1 for DMA Channel 5
DMAC_PERI_LC1_6	0x04d0	W	0x0	Loop Counter 1 for DMA Channel 6
-	-	-	-	reserved
DMAC_PERI_DBG_ST DMAC_PERI_ATU_S	0x0D00	W	0x0	Debug Status Register.
DMAC_PERI_DBG_CMD	0x0D04	W	0x0	Debug Command Register.

DMAC_PERI_DBG_INST0	0x0D08	W	0x0	Debug Instruction-0 Register.
DMAC_PERI_DBG_INST1	0x0D0C	W	0x0	Debug Instruction-1 Register.
DMAC_PERI_CR0	0x0E00	W		Configuration Register 0.
DMAC_PERI_CR1	0x0E04	W		Configuration Register 1.
DMAC_PERI_CR2	0x0E08	W		Configuration Register 2.
DMAC_PERI_CR3	0x0E0C	W		Configuration Register 3.
DMAC_PERI_CR4	0x0E10	W		Configuration Register 4.
DMAC_PERI_CRD_n	0x0E14	W		Configuration Register Dn.
DMAC_PERI_WD	0X0E80	W		Watchdog Register

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 11.4.2 Detail Register Description

#### DMAC\_PERI\_DSR

Address: Operational Base+0x0

DMA Manager Status Register

Bit	Attr	Reset Value	Description
31:10	-	-	Reserved
9	R	0x0	Provides the security status of the DMA manager thread: 0 = DMA manager operates in the Secure state 1 = DMA manager operates in the Non-secure state.
8:4	R	0x0	When the DMA manager thread executes a DMAWFE instruction, it waits for the following event to occur: b00000 = event[0] b00001 = event[1] b00010 = event[2] ... b11111 = event[31].
3:0	R	0x0	The operating state of the DMA manager: b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = Waiting for event b0101-b1110 = reserved b1111 = Faulting.

#### DMAC\_PERI\_DPC

Address: Operational Base+0x4

DMA Program Counter Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Program counter for the DMA manager thread

#### DMAC\_PERI\_INTEN

Address: Operational Base+0x20

Interrupt Enable Register

Bit	Attr	Reset Value	Description

31:0	RW	0x0	<p>Program the appropriate bit to control how the DMAC responds when it executes DMASEV:</p> <p>Bit [N] = 0 If the DMAC executes DMASEV for the event-interrupt resource N then the DMAC signals event N to all of the threads. Set bit [N] to 0 if your system design does not use irq[N] to signal an interrupt request.</p> <p>Bit [N] = 1 If the DMAC executes DMASEV for the event-interrupt resource N then the DMAC sets irq[N] HIGH. Set bit [N] to 1 if your system designer requires irq[N] to signal an interrupt request.</p>
------	----	-----	---

**DMAC\_PERI\_EVENT\_RIS**

Address: Operational Base+0x24

Event-Interrupt Raw Status Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	<p>Returns the status of the event-interrupt resources:</p> <p>Bit [N] = 0 Event N is inactive or irq[N] is LOW.</p> <p>Bit [N] = 1 Event N is active or irq[N] is HIGH.</p>

**DMAC\_PERI\_INTMIS**

Address: Operational Base+0x28

Interrupt Status Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	<p>Provides the status of the interrupts that are active in the DMAC:</p> <p>Bit [N] = 0 Interrupt N is inactive and therefore irq[N] is LOW.</p> <p>Bit [N] = 1 Interrupt N is active and therefore irq[N] is HIGH</p>

**DMAC\_PERI\_INTCLR**

Address: Operational Base+0x2c

Interrupt Clear Register

Bit	Attr	Reset Value	Description
31:0	W	0x0	<p>Controls the clearing of the irq outputs:</p> <p>Bit [N] = 0 The status of irq[N] does not change.</p> <p>Bit [N] = 1 The DMAC sets irq[N] LOW if the INTEN Register programs the DMAC to signal an interrupt.</p> <p>Otherwise, the status of irq[N] does not change.</p>

**DMAC\_PERI\_FSRD**

Address: Operational Base+0x30

Fault Status DMA Manager Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	<p>Provides the fault status of the DMA manager.</p> <p>Read as:</p> <p>0 = the DMA manager thread is not in the Faulting state</p> <p>1 = the DMA manager thread is in the Faulting state.</p>

**DMAC\_PERI\_FSRC**

Address: Operational Base+0x34

Fault Status DMA Channel Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	R	0x0	Each bit provides the fault status of the corresponding channel. Read as: Bit [N] = 0 No fault is present on DMA channel N. Bit [N] = 1 DMA channel N is in the Faulting or Faulting completing state.

**DMAC\_PERI\_FTRD**

Address: Operational Base+0x38

Fault Type DMA Manager Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	-	-	reserved
30	R	0x0	If the DMA manager aborts, this bit indicates if the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface.
29:17	-	-	reserved
16	R	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA manager performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response
15:6	-	-	reserved
5	R	0x0	Indicates if the DMA manager was attempting to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = DMA manager has appropriate security to execute DMAWFE or DMASEV 1 = a DMA manager thread in the Non-secure state attempted to execute either: <ul style="list-style-type: none"><li>• DMAWFE to wait for a secure event</li><li>• DMASEV to create a secure event or secure interrupt</li></ul>
4	R	0x0	Indicates if the DMA manager was attempting to execute DMAGO with inappropriate security permissions: 0 = DMA manager has appropriate security to execute DMAGO 1 = a DMA manager thread in the Non-secure state attempted to execute DMAGO to create a DMA channel operating in the Secure state.
3:2	-	-	reserved
1	R	0x0	Indicates if the DMA manager was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand.
0	R	0x0	Indicates if the DMA manager was attempting to execute an undefined instruction:

			0 = defined instruction 1 = undefined instruction.
--	--	--	---

**DMAC\_PERI\_FTR0~DMAC\_PERI\_FTR6**

Address: Operational Base+0x40

Operational Base+0x44  
 Operational Base+0x48  
 Operational Base+0x4c  
 Operational Base+0x50  
 Operational Base+0x54  
 Operational Base+0x58

Fault Type DMA Channel Register

Bit	Attr	Reset Value	Description
31	R	0x0	Indicates if the DMA channel has locked-up because of resource starvation: 0 = DMA channel has adequate resources 1 = DMA channel has locked-up because of insufficient resources. This fault is an imprecise abort
30	R	0x0	If the DMA channel aborts, this bit indicates if the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface. This fault is an imprecise abort but the bit is only valid when a precise abort occurs.
29:19	-	-	reserved
18	R	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs a data read: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort
17	R	0x0	Indicates the AXI response that the DMAC receives on the BRESP bus, after the DMA channel thread performs a data write: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
16	R	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is a precise abort.
15:14	-	-	reserved
13	R	0x0	Indicates if the MFIFO did not contain the data to enable the DMAC to perform the DMAST: 0 = MFIFO contains all the data to enable the DMAST to complete 1 = previous DMALDs have not put enough data in the MFIFO to enable the DMAST to complete. This fault is a precise abort.
12	R	0x0	Indicates if the MFIFO prevented the DMA

			channel thread from executing DMA LD or DMA ST. Depending on the instruction: DMA LD 0 = MFIFO contains sufficient space 1 = MFIFO is too small to hold the data that DMA LD requires. DMA ST 0 = MFIFO contains sufficient data 1 = MFIFO is too small to store the data to enable DMA ST to complete. This fault is an imprecise abort
11:8	-	-	reserved
7	R	0x0	Indicates if a DMA channel thread, in the Non-secure state, attempts to program the CCRn Register to perform a secure read or secure write: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to perform a secure read or secure write. This fault is a precise abort
6	R	0x0	Indicates if a DMA channel thread, in the Non-secure state, attempts to execute DMA WFP, DMA LD P, DMA ST P, or DMA FLUSH P with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: <ul style="list-style-type: none"><li>• DMA WFP to wait for a secure peripheral</li><li>• DMA LD P or DMA ST P to notify a secure peripheral</li><li>• DMA FLUSH P to flush a secure peripheral.</li></ul> This fault is a precise abort.
5	R	0x0	Indicates if the DMA channel thread attempts to execute DMA WFE or DMA SEV with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: <ul style="list-style-type: none"><li>• DMA WFE to wait for a secure event</li><li>• DMA SEV to create a secure event or secure interrupt.</li></ul> This fault is a precise abort.
4:2	-	-	reserved
1	R	0x0	Indicates if the DMA channel thread was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand. This fault is a precise abort.
0	R	0x0	Indicates if the DMA channel thread was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction. This fault is a precise abort

**DMAC\_PERI\_CSR0~DMAC\_PERI\_CSR6**

Address: Operational Base+0x100

Operational Base+0x108  
 Operational Base+0x110  
 Operational Base+0x118  
 Operational Base+0x120  
 Operational Base+0x128  
 Operational Base+0x130

## Channel Status Registers

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	-	-	reserved
21	R	0x0	The channel non-secure bit provides the security of the DMA channel: 0 = DMA channel operates in the Secure state 1 = DMA channel operates in the Non-secure state
20:16	-	-	reserved
15	R	0x0	When the DMA channel thread executes DMAWFP this bit indicates if the periph operand was set: 0 = DMAWFP executed with the periph operand not set 1 = DMAWFP executed with the periph operand set
14	R	0x0	When the DMA channel thread executes DMAWFP this bit indicates if the burst or single operand were set: 0 = DMAWFP executed with the single operand set 1 = DMAWFP executed with the burst operand set.
13:9	-	-	reserved
8:4	R	0x0	If the DMA channel is in the Waiting for event state or the Waiting for peripheral state then these bits indicate the event or peripheral number that the channel is waiting for: b00000 = DMA channel is waiting for event, or peripheral, 0 b00001 = DMA channel is waiting for event, or peripheral, 1 b00010 = DMA channel is waiting for event, or peripheral, 2 . . . b11111 = DMA channel is waiting for event, or peripheral, 31
3:0	R	0x0	The channel status encoding is: b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = Waiting for event b0101 = At barrier b0110 = reserved b0111 = Waiting for peripheral

		b1000 = Killing b1001 = Completing b1010-b1101 = reserved b1110 = Faulting completing b1111 = Faulting
--	--	--

**DMAC\_PERI\_CPC0~DMAC\_PERI\_CPC6**

Address: Operational Base+0x104

Operational Base+0x10c  
Operational Base+0x114  
Operational Base+0x11c  
Operational Base+0x124  
Operational Base+0x12c  
Operational Base+0x134

Channel Program Counter Registers

Bit	Attr	Reset Value	Description
31:0	R	0x0	Program counter for the DMA channel n thread

**DMAC\_PERI\_SAR0~DMAC\_PERI\_SAR6**

Address: Operational Base+0x400

Operational Base+0x420  
Operational Base+0x440  
Operational Base+0x460  
Operational Base+0x480  
Operational Base+0x4a0  
Operational Base+0x4c0

**Source Address Registers**

Bit	Attr	Reset Value	Description
31:0	R	0x0	Address of the source data for DMA channel n

**DMAC\_PERI\_DAR0~DMAC\_PERI\_DAR5**

Address: Operational Base+0x404

Operational Base+0x424  
Operational Base+0x444  
Operational Base+0x464  
Operational Base+0x484  
Operational Base+0x4a4

DestinationAddress Registers

Bit	Attr	Reset Value	Description
31:0	R	0x0	Address of the Destinationdata for DMA channel n

**DMAC\_PERI\_CCR0~DMAC\_PERI\_CCR6**

Address: Operational Base+0x408

Operational Base+0x428  
Operational Base+0x448  
Operational Base+0x468  
Operational Base+0x488  
Operational Base+0x4a8  
Operational Base+0x4c8

Channel Control Registers

Bit	Attr	Reset Value	Description
31:28	-	-	reserved
27:25	R	0x0	Programs the state of AWCACHE[3,1:0]a when the DMAC writes the destination data. Bit [27] 0 = AWCACHE[3] is LOW 1 = AWCACHE[3] is HIGH.

			Bit [26] 0 = AWCACHE[1] is LOW 1 = AWCACHE[1] is HIGH. Bit [25] 0 = AWCACHE[0] is LOW 1 = AWCACHE[0] is HIGH
24:22	R	0x0	Programs the state of AWPROT[2:0]a when the DMAC writes the destination data. Bit [24] 0 = AWPROT[2] is LOW 1 = AWPROT[2] is HIGH. Bit [23] 0 = AWPROT[1] is LOW 1 = AWPROT[1] is HIGH. Bit [22] 0 = AWPROT[0] is LOW 1 = AWPROT[0] is HIGH
21:18	R	0x0	For each burst, these bits program the number of data transfers that the DMAC performs when it writes the destination data: b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers. The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size
17:15	R	0x0	For each beat within a burst, it programs the number of bytes that the DMAC writes to the destination: b000 = writes 1 byte per beat b001 = writes 2 bytes per beat b010 = writes 4 bytes per beat b011 = writes 8 bytes per beat b100 = writes 16 bytes per beat b101-b111 = reserved. The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size.
14	R	0x0	Programs the burst type that the DMAC performs when it writes the destination data: 0 = Fixed-address burst. The DMAC signals AWBURST[0] LOW. 1 = Incrementing-address burst. The DMAC signals AWBURST[0] HIGH.
13:11	R	0x0	Set the bits to control the state of ARCACHE[2:0]a when the DMAC reads the source data. Bit [13] 0 = ARCACHE[2] is LOW 1 = ARCACHE[2] is HIGH. Bit [12] 0 = ARCACHE[1] is LOW 1 = ARCACHE[1] is HIGH. Bit [11] 0 = ARCACHE[0] is LOW 1 = ARCACHE[0] is HIGH.
10:8	R	0x0	Programs the state of ARPROT[2:0]a when the DMAC reads the source data. Bit [10] 0 = ARPROT[2] is LOW

			1 = ARPROT[2] is HIGH. Bit [9] 0 = ARPROT[1] is LOW 1 = ARPROT[1] is HIGH. Bit [8] 0 = ARPROT[0] is LOW 1 = ARPROT[0] is HIGH.
7:4	R	0x0	For each burst, these bits program the number of data transfers that the DMAC performs when it reads the source data: b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers. The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size
3:1	R	0x0	For each beat within a burst, it programs the number of bytes that the DMAC reads from the source: b000 = reads 1 byte per beat b001 = reads 2 bytes per beat b010 = reads 4 bytes per beat b011 = reads 8 bytes per beat b100 = reads 16 bytes per beat b101-b111 = reserved. The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size
0	R	0x0	Programs the burst type that the DMAC performs when it reads the source data: 0 = Fixed-address burst. The DMAC signals ARBURST[0] LOW. 1 = Incrementing-address burst. The DMAC signals ARBURST[0] HIGH

**DMAC\_PERI\_LC0\_0~DMAC\_PERI\_LC0\_6**

Address: Operational Base+0x40c  
                   Operational Base+0x42c  
                   Operational Base+0x44c  
                   Operational Base+0x46c  
                   Operational Base+0x48c  
                   Operational Base+0x4ac  
                   Operational Base+0x4cc

Loop Counter 0 Registers

Bit	Attr	Reset Value	Description
31:8	-	-	reserved
7:0	R	0x0	Loop counter 0 iterations

**DMAC\_PERI\_LC1\_0~DMAC\_PERI\_LC1\_6**

Address: Operational Base+0x410  
                   Operational Base+0x430  
                   Operational Base+0x450  
                   Operational Base+0x470

Operational Base+0x490  
 Operational Base+0x4b0  
 Operational Base+0x4e0

## Loop Counter 1 Registers

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	-	-	reserved
7:0	R	0x0	Loop counter 1 iterations

**DMAC\_PERI\_DBGSTATUS**

Address: Operational Base+0xd00

Debug Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	-	-	reserved
1:0	R	0x0	The debug encoding is as follows: b00 = execute the instruction that the DBGINST [1:0] Registers contain b01 = reserved b10 = reserved b11 = reserved.

**DMAC\_PERI\_DBGCMD**

Address: Operational Base+0xd04

Debug Command Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	-	-	reserved
1:0	W	0x0	The debug encoding is as follows: b00 = execute the instruction that the DBGINST [1:0] Registers contain b01 = reserved b10 = reserved b11 = reserved

**DMAC\_PERI\_DBGINST0**

Address: Operational Base+0xd08

Debug Instruction-0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	W	0x0	Instruction byte 1
23:16	W	0x0	Instruction byte 0
17:11	-	-	reserved
10:8	W	0x0	DMA channel number: b000 = DMA channel 0 b001 = DMA channel 1 b010 = DMA channel 2 ... b111 = DMA channel 7
7:1	-	-	reserved
0	W	0x0	The debug thread encoding is as follows: 0 = DMA manager thread 1 = DMA channel.

**DMAC\_PERI\_DBGINST1**

Address: Operational Base+0xd0c

Debug Instruction-1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	W	0x0	Instruction byte 5

23:16	W	0x0	Instruction byte 4
15:8	W	0x0	Instruction byte 3
7:0	W	0x0	Instruction byte 2

**DMAC\_PERI\_CRO**

Address: Operational Base+0xe00

Configuration Register 0

Bit	Attr	Reset Value	Description
31:22	-	-	reserved
21:17	R	0x2	<p>Number of interrupt outputs that the DMAC provides:</p> <p>b00000 = 1 interrupt output, irq[0]      b00001 = 2 interrupt outputs, irq[1:0]      b00010 = 3 interrupt outputs, irq[2:0]</p> <p>.</p> <p>.</p> <p>b11111 = 32 interrupt outputs, irq[31:0].</p>
16:12	R	0x7	<p>Number of peripheral request interfaces that the DMAC provides:</p> <p>b00000 = 1 peripheral request interface      b00001 = 2 peripheral request interfaces      b00010 = 3 peripheral request interfaces</p> <p>.</p> <p>.</p> <p>b11111 = 32 peripheral request interfaces.</p>
11:7	-	-	reserved
6:4	R	0x5	<p>Number of DMA channels that the DMAC supports:</p> <p>b000 = 1 DMA channel      b001 = 2 DMA channels      b010 = 3 DMA channels</p> <p>.</p> <p>.</p> <p>b111 = 8 DMA channels.</p>
3	-	-	reserved
2	R	0x0	<p>Indicates the status of the boot_manager_ns signal when the DMAC exited from reset:</p> <p>0 = boot_manager_ns was LOW      1 = boot_manager_ns was HIGH.</p>
1	R	0x0	<p>Indicates the status of the boot_from_pc signal when the DMAC exited from reset:</p> <p>0 = boot_from_pc was LOW      1 = boot_from_pc was HIGH</p>
0	R	0x1	<p>Supports peripheral requests:</p> <p>0 = the DMAC does not provide a peripheral request interface      1 = the DMAC provides the number of peripheral request interfaces that the num_periph_req field specifies.</p>

**DMAC\_PERI\_CR1**

Address: Operational Base+0xe04

Configuration Register 1

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:8	-	-	reserved
7:4	R	0x5	[7:4] num_i-cache_lines Number of i-cache lines: b0000 = 1 i-cache line b0001 = 2 i-cache lines b0010 = 3 i-cache lines ... b1111 = 16 i-cache lines.
3	-	-	reserved
2:0	R	0x7	The length of an i-cache line: b000-b001 = reserved b010 = 4 bytes b011 = 8 bytes b100 = 16 bytes b101 = 32 bytes b110-b111 = reserved

**DMAC\_PERI\_CR2**

Address: Operational Base+0xe08

Configuration Register 2

Bit	Attr	Reset Value	Description
31:0	R	0x0	Provides the value of boot_addr[31:0] when the DMAC exited from reset

**DMAC\_PERI\_CR3**

Address: Operational Base+0xe0c

Configuration Register 3

Bit	Attr	Reset Value	Description
31:0	R	0x0	Provides the security state of an event-interrupt resource: Bit [N] = 0 Assigns event<N> or irq[N] to the Secure state. Bit [N] = 1 Assigns event<N> or irq[N] to the Non-secure state.

**DMAC\_PERI\_CR4**

Address: Operational Base+0xe10

Configuration Register 4

Bit	Attr	Reset Value	Description
31:0	R	0x6	Provides the security state of the peripheral request interfaces: Bit [N] = 0 Assigns peripheral request interface N to the Secure state. Bit [N] = 1 Assigns peripheral request interface N to the Non-secure state

**DMAC\_PERI\_CRDn**

Address: Operational Base+0xe14

DMA Configuration Register

Bit	Attr	Reset Value	Description
31:30	-	-	reserved
29:20	R	0x20	The number of lines that the data buffer contains: b000000000 = 1 line b000000001 = 2 lines ... b111111111 = 1024 lines

19:16	R	0x9	The depth of the read queue: b0000 = 1 line b0001 = 2 lines . . . b1111 = 16 lines.
15	-	-	reserved
14:12	R	0x4	Read issuing capability that programs the number of outstanding read transactions: b000 = 1 b001 = 2 ... b111 = 8
11:8	R	0x7	The depth of the write queue: b0000 = 1 line b0001 = 2 lines . . . b1111 = 16 lines.
7	-	-	reserved
6:4	R	0x3	Write issuing capability that programs the number of outstanding write transactions: b000 = 1 b001 = 2 . . . b111 = 8
3	-	-	reserved
2:0		0x3	The data bus width of the AXI interface: b000 = reserved b001 = reserved b010 = 32-bit b011 = 64-bit b100 = 128-bit b101-b111 = reserved.

**DMAC\_PERI\_WD**

Address: Operational Base+0xe80

DMA Watchdog Register

Bit	Attr	Reset Value	Description
31:1	-	-	reserved
0	RW	0x0	Controls how the DMAC responds when it detects a lock-up condition: 0 = the DMAC aborts all of the contributing DMA channels and sets irq_abort HIGH 1 = the DMAC sets irq_abort HIGH.

**11.5 Timing Diagram**

Please refer to chapter 10.5 for the similar description.

**11.6 Interface Description**

DMAC\_PERI has the following tie-off signals. It can be configured by GRF register. (Please refer to the chapter to find how to configure)

**DMAC\_PERI**

interface	Reset value	Control source
boot_addr	0x0	GRF
boot_from_pc	0x0	GRF
boot_manager_ns	0x0	GRF
boot_irq_ns	0xf	GRF
boot_periph_ns	0xfffff	GRF

**boot\_addr**

Configures the address location that contains the first instruction the DMAC executes, when it exits from reset.

**boot\_from\_pc**

Controls the location in which the DMAC executes its initial instruction, after it exits from reset:

- 0 = DMAC waits for an instruction from either APB interface
- 1 = DMA manager thread executes the instruction that is located at the address that

**boot\_manager\_ns**

When the DMAC exits from reset, this signal controls the security state of the DMA manager thread:

- 0 = assigns DMA manager to the Secure state
- 1 = assigns DMA manager to the Non-secure state.

**boot\_irq\_ns**

Controls the security state of an event-interrupt resource, when the DMAC exits from reset:

- boot\_irq\_ns[x] is LOW  
The DMAC assigns event<x> or irq[x] to the Secure state.
- boot\_irq\_ns[x] is HIGH  
The DMAC assigns event<x> or irq[x] to the Non-secure state.

**boot\_periph\_ns**

Controls the security state of a peripheral request interface, when the DMAC exits from reset:

- boot\_periph\_ns[x] is LOW  
The DMAC assigns peripheral request interface x to the Secure state.
- boot\_periph\_ns[x] is HIGH  
The DMAC assigns peripheral request interface x to the Non-secure state.

## 11.7 Application Notes

Please refer to chapter 10.7 for the similar description.

## Chapter 12 Generic Interrupt Controller (GIC)

### 12.1 Overview

The generic interrupt controller (GIC400) in this device has two interfaces, the distributor interface connects to the interrupt source, and the CPU interface connects to Cortex-A17.

It supports the following features:

- Supports 160 hardware interrupt inputs
- Masking of any interrupts
- Prioritization of interrupts
- Distribution of the interrupts to the target Cortex-A17 processor(s)
- Generation of interrupts by software
- Supports Security Extensions

### 12.2 Block Diagram

The generic interrupt controller comprises with:

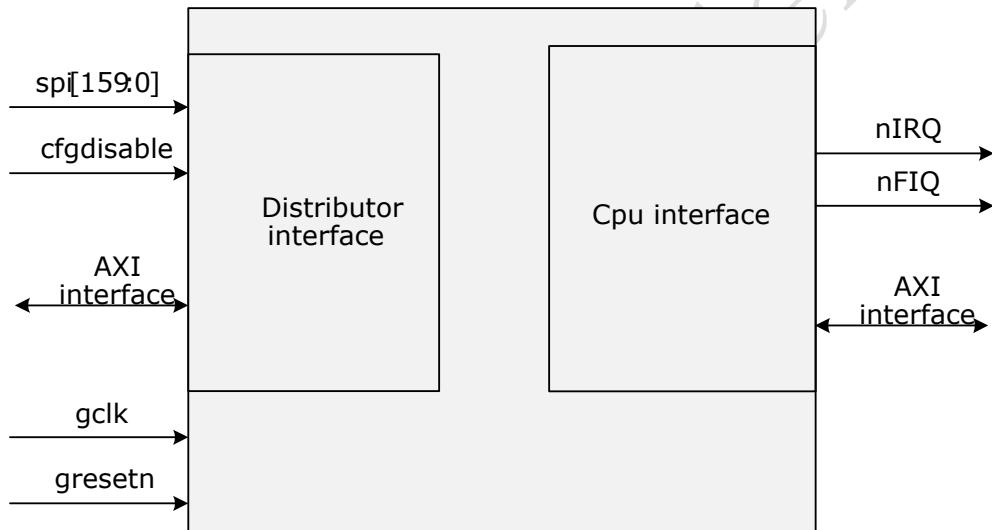


Fig. 12-1 Block Diagram

### 12.3 Function Description

Please refer to the document IHI0048B\_gic\_architecture\_specification.pdf for the cpu detail description.

## Chapter 13 Dynamic Memory Interface (DMC)

### 13.1 Overview

The DMC includes two section: dynamic ram protocol controller(PCTL) and phy controller (PHYCTL).

The PCTL SoC application bus interface supports a lowest-latency native application interface (NIF). To maximize data transfer efficiency, NIF commands transfer data without flow control. To simplify command processing, the NIF accepts addresses in rank, bank, row, column format.

The PHYCTL provides control features to ease the customer implementation of digitally controlled features of the PHY such as initialization, DQS gate training, and programmable configuration controls. The PHYCTL has built-in self test features to provide support for production testing of the compatible PHY. It also provides a DFI 2.1 interface to the PHY.

The DMC supports the following features:

- Complete, integrated, single-vendor DDR3, DDR3L, LPDDR2, LPDDR3 solution
- DFI 2.1 interface compatibility
- Up to 1066 Mbps in 1:1 frequency ratio, using a 533MHz controller clock and 533MHz memory clock
- Dual channel, each channel up to 32 bits, totally support 64 bits data width
- Support for x8, x16, and x32 memories
- Each channel has separately controller and PHY
- Up to 2 memory ranks for each channel; devices within a rank tie to a common chip select
- Up to 8 open memory banks, maximum of eight per rank
- Per-NIF transaction controllable bank management policies: open-page, close-page
- Low area, low power architecture with minimal buffering on the data, avoiding duplication of storage resources within the system
- PCTL NIF slave interface facilitates easy integration with an external scheduler or standard on-chip buses
- Efficient DDR protocol implementation with in-order column (Read and Write) commands and out-of-order Activate and Precharge commands
- Three clock cycles best case command latency (best case is when a command is to an open page and the shift array in the PCTL is empty).
- 1T or 2T memory command timing
- Automatic clock stop, power-down and self-refresh entry and exit.
- Clock stop is LPDDR2/LPDDR3 only
- Software and hardware driven self-refresh entry and exit
- Programmable memory initialization
- Partial population of memories, where not all DDR byte lanes are populated with memory chips
- Programmable per rank memory ODT(On-Die Termination) support for reads and writes
- APB interface for controller software-accessible registers
- Programmable data training interface
- Assists in training of the data eye of the memory channel
- Provides a method for testing large sections of memory
- Support for industry standard UDIMMs (Unbuffered DIMMs) and RDIMMs (Registered DIMMs)
- Automatic DQS gate training and drift compensation
- At-speed built-in-self-test(BIST) loopback testing on both the address and data channels for DDR PHYs
- PHY control and configuration registers
- Support 2G memory wrap function

## 13.2 Block Diagram

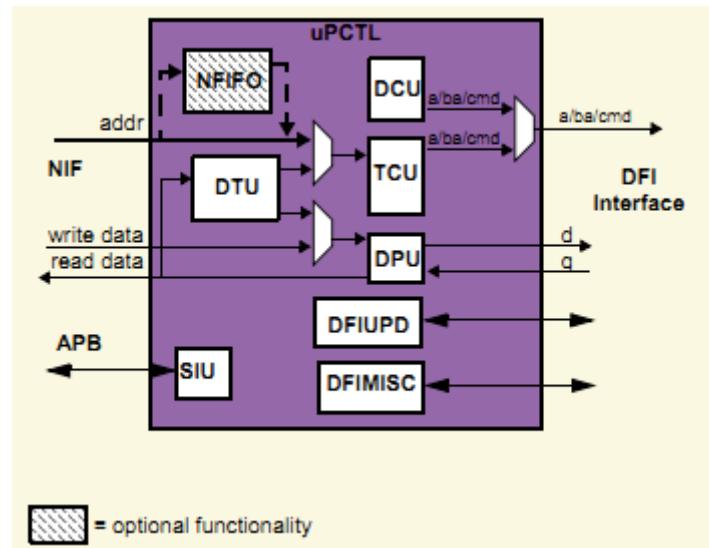


Fig. 13-1 Protocol controller architecture

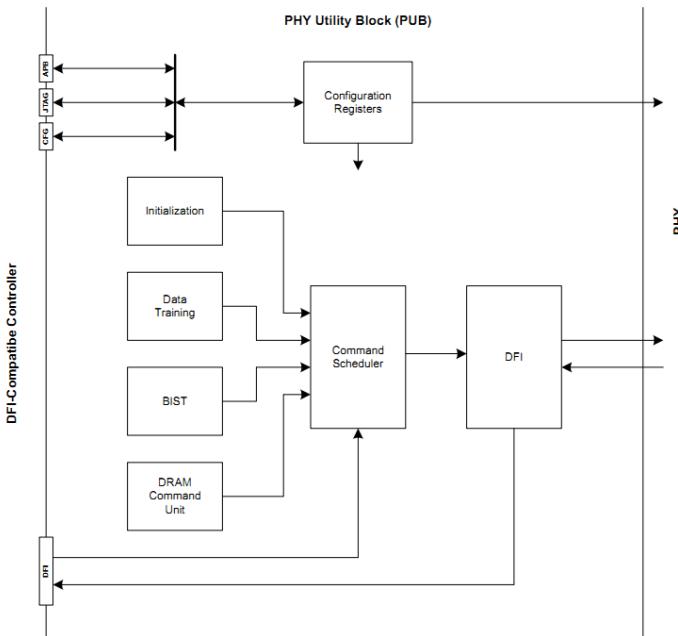


Fig. 13-2 PHY controller architecture

## 13.3 Function description

PCTL operations are defined in terms of the current state of the Operational State Machine. Software can move PCTL in any of the operational states by issuing commands via the SCTL register. Transitions from one operational state to the other occur pass through a “transitional” state. Transitional states are exited automatically by the PCTL after all the necessary actions required to change operational state have been completed. The current operational state of PCTL is reported by the STAT register and is also available from the p\_ctl\_stat output.

PCTL supports the following operational states:

- **Init\_mem** - This state is the default state entered after reset. All writable registers can be programmed. While in this state software can program PCTL and initialize the PHY and the memories. The memories are not refreshed and data that has previously been written to the memories may be lost as a result. The Init\_mem state is also used when it is desirable to stop any automatic PCTL function that directly affects the memories, like Power Down and Refresh, or when software reset of the memory subsystem has to be executed.

- Config - This state is used to suspend temporarily the normal NIF traffic and allow software to reprogram PCTL and memories if necessary, while still keeping active the periodic generation of Refresh cycles to the memories. Power Down entry and exit sequences are possible while in Config state.
- Access - This is the operational state where NIF transactions are accepted by the PCTL and converted into memory read and writes. None of the registers can be programmed except SCFG, SCTL, ECCCLR and DTU\* registers.
- Low\_power - Memories are in self refresh mode. The PCTL does not generate refresh cycles while in this state.

Access and Low\_power states can also be entered and exited by the hardware low power signals ( $c_*$ ). In case of conflicting software and hardware low-power commands, the resulting operational state taken by the controller can be either one of the two conflicting requests.

Figure 13-3 illustrates the operational and transitional states.

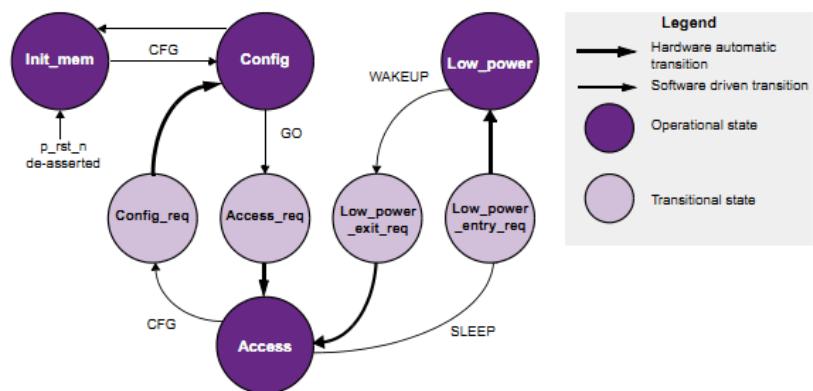


Fig. 13-3 Protocol controller architecture

The PHYCTL provides control features to ease the customer implementation of digitally controlled PHY features such as read DQS training, data eye training, output impedance calibration, and so on. The PHTCTL has built-in self test features to provide support for production testing of PHY. It also provides a DFI 2.1 interface to the PHY. The PHYCTL performs, in sequence, various tasks required by the PHY before it can commence normal DDR operations. SDRAM memory read/write access through the DDR PHY is primarily through a DFI 2.1 interface on the PHYCTL. Therefore, the memory controller used with the PHY must be DFI 2.1 compatible.

Access to the PHYCTL internal control features and registers is through a dedicated configuration port, which can be either APB or CFG (generic configuration interface). An optional JTAG interface can also be compiled in as an additional second configuration port to co-exist with either the APB or CFG main configuration ports. The PHYCTL is driven off two clocks, the controller clock ( $ctl\_clk$ ) and the configuration clock  $pclk$  for an APB interface.

The controller clock is the same clock driving the memory controller and will be the same frequency as the SDRAM clock ( $ck$ ). The configuration clock can run at a frequency equal to or less than the controller clock. The configuration clock drives all non-DDR timing logic, such as configuration registers, PHY initialization, output impedance, and so on.

## 13.4 DDR PHY

### 13.4.1 DDR PHY Overview

In order to facilitate robust system timing and ease of use, DMC interface and control architecture utilizes a mixture of soft-IP and hard-IP design elements. The main control logic (Memory Controller) is supplied as soft-IP. The PHY is comprised of hard-IP components that include double-data rate Interface Timing Modules(ITM), input and output path DLLs, and

application-specific SSTL I/Os.

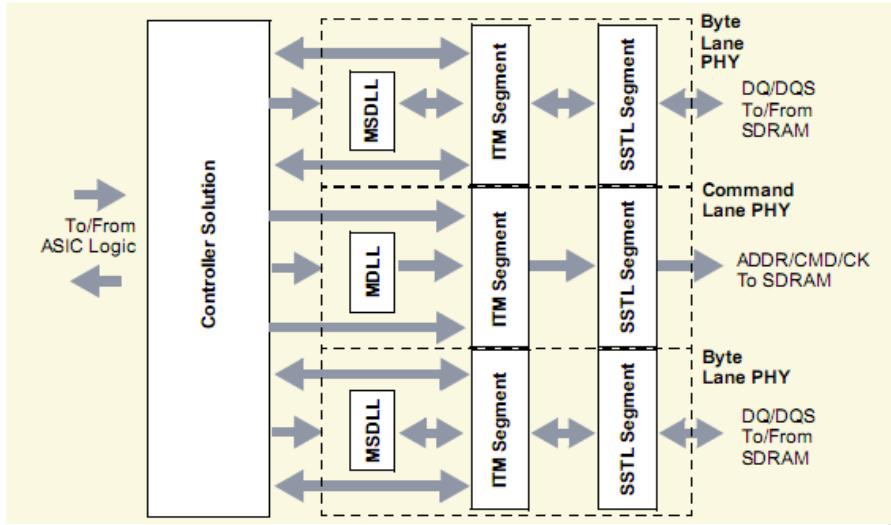


Fig. 13-4 DDR PHY architecture

In order to maximize system timing margins on the command/write path, inputs to the SDRAM are provided with the clock or data strobe centered in the associated data eye. The ITM components perform timing translation for the various signal groups of the interface. The hardened ITM approach ensures minimal pin to pin skew while allowing optimal circuit design for drive and capture circuitry. A DLL is utilized to facilitate the clock centering. In the Command Lane, a master DLL (MDLL) is utilized. In the Byte Lane, the master portion of a master/slave DLL macrocell (MSDLL) is utilized.

On the read path, read data from the SDRAM is arriving from the SDRAM edge aligned with the data strobes. In order to maintain maximum system timing margins on the input path, the data strobes are translated to the center of the data eye. The MSDLLE macrocell associated with each Byte Lane contains a master DLL and 2 slave DLLs (mirror delay lines). The slave DLL portion of the MSDLLE is utilized to facilitate the clock centering. DQS and DQS<sub>b</sub> strobe inputs each utilize one of these slave DLL functions. The captured double data rate inputs are then converted to single data rate and passed onto the DDR Controller RTL logic. The ITM facilitates both data capture and DDR to SDR conversion.

The physical interface between the DDR controller and DDR SDRAMs uses DDR-specific SSTL I/O buffers with programmable on-die termination (ODT). These I/Os operate at either 1.8V for LPDDR/DDR2 interfacing (SSTL\_18).

DMC interface and control architecture follows a common signal grouping philosophy. A Byte Lane is a complete eight-bit data unit consisting of the associated DQ, DM, and DQS/DQS<sub>b</sub> signals. A 32-bit system would consist of four Byte Lanes. A Command Lane is a complete command and address unit including also clock signals. There would normally be only one Command Lane in a particular DDR SDRAM interface. All clock and data signals relative to a Lane, either Byte or Command, are isolated to within that Lane only. Timing critical clock and data signals do not traverse between Lanes. Implementation of a memory interface involves placing the Command Lane components, placing the Byte Lane components, and standard synthesis/place and route to complete the design.

Each SSTL cell communicating with the SDRAM has an associated ITM component. The ITM library consists of individual components designed specifically for signal groups of address and command, data & data mask, and data strobes. In order to ensure low pin to pin skews and facilitate ease of implementation, the ITM components are tileable. DLL output clock distribution is embedded within the ITM components.

### 13.4.2 Lane-Based Architecture

#### Byte Lane PHY

The data bus interface to the external memory is organized into self-contained units referred to herein as Byte Lanes. The external memory components are designed to support Byte Lanes for optimal system timing. The partitioning of the data word into discrete Byte Lanes allows pin to pin skew to be managed across a much smaller group of signals than would typically be required.

All components of the Byte Lane PHY are designed to permit connectivity by abutment. The ITM connects by abutment to the SSTL I/O, and the DLL connects by abutment to the ITM.

The SDRAM contains data strobes associated with each 8 bits of data and there is a timing skew allowance between the main clock signal to the SDRAM and its data strobe inputs during a Write command (tDQSS). 8bit memory components provide a single DQS.

A Byte Lane consists of the following I/O slots:

- 8 data bits (DQ)
- data strobe bits (DQS / DQS\_b)
- 1 data mask bit (DM)
- I/O power and ground cells
- Core power and ground cells

Each functional I/O slot has an associated ITM module, including DQ, DM, and DQS/DQS\_b. The ITMs provide a mechanism for monitoring read timing drift, which can be used to adjust timing to maintain optimum system margins. Drift analysis and compensation is performed by the controller on a per Byte Lane basis. The ITM components contain the functions to monitor DQS drift and permit timing adjustment, the controller provides the analysis and control for these functions. These functions operate dynamically for each data bit of every user-issued Read command. There are no overhead penalties in channel bandwidth or utilization incurred by the use of these functions.

The memory interface (PHY) architecture is based on the concept of independent, but related, signal groups to provide the highest level of system timing performance. In order to maintain robust system timing, all clock and data signals relevant to a Byte Lane remain within that Byte Lane. These signals are not shared between other Byte Lanes or between a Byte Lane and a Command Lane. Alternate approaches require clock distribution networks that span the full length of the interface including all address, command, and data signals. These large clock distribution networks are difficult for the user to design and implement, and add an additional component of pin to pin skew to the critical timing budget.

A DLL macrocell (MSDLL) consisting of a master DLL and 2 slave DLLs (mirror delay lines) is utilized at each Byte Lane to facilitate optimal PHY timing for drive and capture of DDR data streams, and allows the Lanes to be independent. The master DLL section provides outputs for DDR data stream creation to the SDRAMs and acts as a reference for the slave delay line sections. The slave delay line sections translate the incoming DQS/DQS\_b into the center of the read data eye to maximize read system timing margins.

The user is permitted to fine tune the relationship of the DQS and DQ signals to maximize read system timing margin. The DLL includes adjustability of the slave delay lines for the DQS and DQS\_b signals, which provide byte-wide timing adjustments. The ITMs include adjustability of the read DQS/DQS\_b strobe timing, which provides byte-wide timing adjustments. The ITMs include adjustability of the read DQ signal timing, which provides per-bit timing adjustability. To permit Lane-independent timing adjustments, DLL adjustment bits are provided by the controller per Byte Lane and ITM adjustment bits are provided per bit.

DMC interface and control solution allows memory systems with a word width narrower than the design. Our system is designed with a 32 bit data width and it can then be utilized with either 16 bit or 32 bit memory systems. The controller contains register settings to allow the desired operational mode to be set in the final device.

The DDR-specific SSTL I/Os include programmable ODT and output impedance selection. The ODT and output impedances can be dynamically calibrated to compensate for variations in voltage and temperature. The ODT feature can be disabled by the controller.

When ODT is enabled by the controller, the SSTL I/O automatically enables its internal ODT circuitry when in input mode and disable this circuitry when in output mode, as determined by the output enable signal. The initial programming and subsequent calibration of the ODT and output impedance is achieved through the use of an impedance control loop that can be triggered to calibrate the ODT and output impedance values at the I/Os based on the desired impedance value when compared to an precision external resistor. All the necessary pieces of the impedance control loop are included in the SSTL I/O library.

There are four Byte Lanes in our chip of 32 bit memory system.

### **Command Lane PHY**

The control and address interface to the external memory is organized into a self-contained unit referred to herein as a Command Lane. DMC interface contains a single Command Lane and four Byte Lanes.

All components of the Command Lane PHY are designed to permit connectivity by abutment. The ITM connects by abutment to the SSTL I/O, and the DLL connects by abutment to the ITM.

A typical Command Lane consists of the following I/O slots:

- Memory clocks (CK/CK\_b)
- Command signals (RAS\_b, CAS\_b, WE\_b)
- 1 or more clock enable (CKE)
- 1 or more on-die termination (ODT)
- chip select (CS\_b)
- bank address (BA)
- 16 row/column address (A)
- I/O power and ground cells
- Core power and ground cells

The system clock input is used to provide the source clock for the memory interface. Memory controller supports 2 SDRAM ranks. There is one CKE, ODT, and CS\_b signal provided for each rank.

Each functional I/O slot has an associated ITM module, with exception of the system clock input. A master DLL (MDLL) is utilized with the Command Lane to facilitate optimal PHY timing for drive of DDR data streams, and allows the Lane to be independent. The DLL macrocells provide two 0 degree phase outputs, one which can be used to drive the controller logic. The Command Lane MDLL is used for this purpose.

To permit Lane-independent timing adjustments, DLL and ITM adjustment bits are provided by the controller separately for Command and Byte Lanes.

#### **13.4.3 Master DLL(MDLL)**

Master DLL for DDR2, and LPDDR applications is a Delay Locked Loop that takes an input reference clock (clk\_in) and generates four clock outputs, each delayed in quarter clock cycle (90°) increments. These four clock phases (clk\_0, clk\_90, clk\_180, clk\_270) can be generated with very high accuracy and low jitter across a wide range of frequencies.

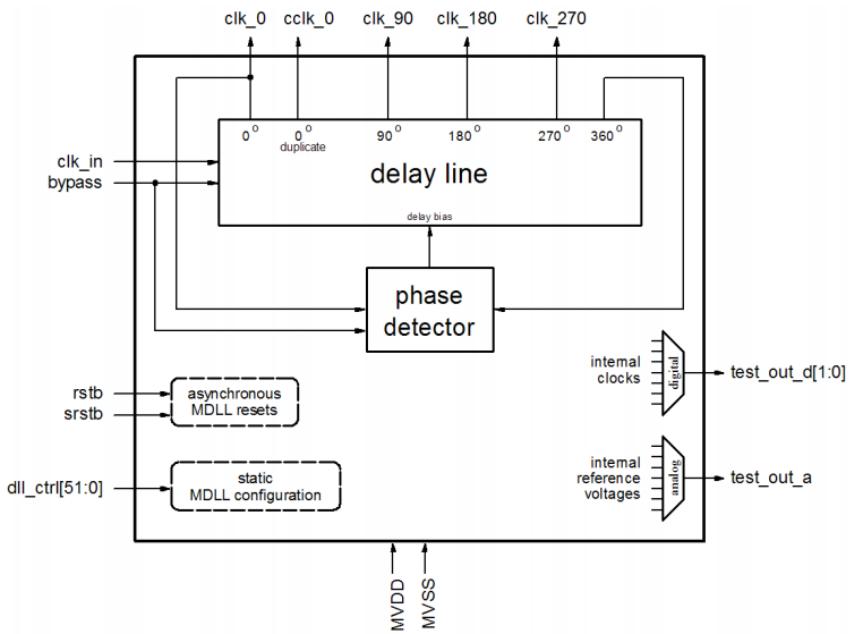


Fig. 13-5 DDR PHY master DLL architecture diagram

A number of test modes and configuration settings are included:

- A bypass mode shuts down all analog circuitry, and directly buffers the input clock and strobes with appropriate delays and inversions to the output clocks and strobes. This mode can be used for low speed functional or IDDQ testing.
- A digital test output (test\_out\_d) provides direct observability of several internal reference clock and timing nodes.
- An analog test output (test\_out\_a) provides direct observability of several internal reference voltages.

### Master DLL Control for Trim and Test

The performance and testing of the MDLL can be accessed through the dll\_ctrl bus.

Table 13-1 DDR PHYtrim and test MDLL control

Static Input	Field	Description
dll_ctrl	[1:0]	Reserved
	[4:2] ipump_trm[2:0]	Charge pump current trim
	[5]	Test control enable for analog and digital test outputs
	[8:6] test_ctrl_d[2:0]	Digital test control. Selects the digital signal to be viewed at the digital test output
	[10:9] test_ctrl_a[1:0]	Analog test control. Selects the analog signal to be viewed at the analog test output
	[11]	Reserved
	[14:12] bias_trm[2:0]	Bias generator frequency trim
	[19,15] fdtrm[1:0]	Bypass mode fixed delay trim
	[22:20] bias_trm[6:4]	Bias generator control voltage trim
	[23]	Bypass frequency select
	[28:24]	Reserved
	[29]	Reserved
	[37:30]	Reserved
	[43:38] fb_trm[5:0]	Feedback delay adjust
	[49:44]	Reserved

	[50]	test_hizb_a	Analog test output tri-stated control
	[51]		Reserved

Charge Pump Current Trim:

Table 13-2 charge pump current trim in dll\_ctrl

Field	Setting	Function	Suggested Default
ipump_trm[2:0]	000	Maximum current	000
	111	Minimum current	

Digital Test Control:

Table 13-3 DLL digital test control in dll\_ctrl

test_ctrl_en	test_ctrl_d[2:0]	Function	Suggested Default
0	xxx	digital test outputs disabled (drive '0')	0,000
1	000	0° output clock (clk_0)	
1	001	90° output clock (clk_90)	
1	010	180° output clock (clk_180)	
1	011	270° output clock (clk_270)	
1	100	360° internal clock (clk_360_int)	
1	101	Speed-up pulse (spdup)	
1	110	Slow-down pulse (slwdn)	
1	111	Asic output clock (cclk_0)	

Analog Test Control:

Table 13-4 DLL analog test control in dll\_ctrl

test_hizb_a	test_ctrl_en	test_ctrl_a [1:0]	Function	Suggested Default
0	x	xx	Tri-state	0,0,00
1	0	xx	MVSS	
1	1	00	Filter output (Vc)	
1	1	01	Replica bias output for NMOS (Vbn)	
1	1	10	Replica bias output for PMOS (Vbp)	
1	1	11	MVDD	

Bias Generator Trim:

The bias generator trim capability can be used to adjust the behavior of the bias voltages being supplied to the delay line. Characteristics of the DLL that may warrant an adjustment of this trim value include the inability to lock due to a slow clock (suggest decreasing Vc adjust), inability to lock due to fast clock (suggest increasing Vc adjust) and increase noise margin on bias voltages (suggest decreasing Fmax adjust). The bit fields described in the following table can be set to any value between 000(binary) and 111(binary).

Table 13-5 bias generator trim in dll\_ctrl

Field	Setting	Function	Suggested Default
bias_trm[2:0]	000	Fmax trim: minimum adjust	111
	111	Fmax trim: maximum adjust	
bias_trm[6:4]	000	Vc level trim: minimum adjust	011
	111	Vc level trim: maximum adjust	

Feedback Trim:

The feedback trim capability can be used in the event that an adjustment is desired in the

phase detector feedback of the DLL. Characteristics of the DLL that may warrant an adjustment of this trim value include non-optimal phase alignment. The lower 3 bits (2:0) are used for feed-back delay trimming and the upper 3 bits (5:3) are used for feed-forward delay trimming. The feed-back trimming is used to decrease total delay, decreasing the amount of delay between phase outputs. The feed-forward trimming is used to increase total delay, increasing the amount of delay between phase outputs. For each 3-bit field, the inputs can be set to any value between 000(binary) and 111(binary).

Table 13-6 MDLL feedback trim in dll\_ctrl

Field	Setting	Function	Suggested Default
fb_trm[5:3] (feed-forward path)	000	Minimum additional delay	000
	111	Maximum additional delay	
fb_trm[2:0] (feed-back path)	000	Minimum additional delay	000
	111	Maximum additional delay	

### Bypass Mode

The DLL has a bypass mode which allows phased clocks to be generated with analog locking circuitry disabled. This mode may be used for low-speed functional testing and for IDDq testing. Bypass mode can also be used when operating with LPDDR SDRAMs. When bypass mode is enabled, all analog circuitry is disabled, and all static current paths are shut down.

Bypass mode has two settings for the clk\_90 delay to optimize it for two different frequency ranges.

Table 13-7 MDLL bypass mode frequency range in dll\_ctrl

Field	Setting	Function	Suggested Default
bps200	0	0 to 100MHz	0
	1	0 to 200MHz	

It is also possible to trim the 90-degree delay using the fdtrm control bits.

Table 13-8 fdtrm control bits in dll\_ctrl

Field	Setting	Function	Suggested Default
fdtrm[1:0]	00	nominal delay	00
	01	nominal delay - 10%	
	10	nominal delay + 10%	
	11	nominal delay + 20%	

### 13.4.4 Master-Slave DLL(MSDLL)

Master-Slave DLL for DDR2, and LPDDR applications is an integrated Delay Locked Loop and a pair of slave delays. The Delay Locked Loop (DLL) takes an input reference clock (clk\_in), and generates four clock outputs, each delayed in quarter clock cycle (90°) increments. These four clock phases (clk\_0, clk\_90, clk\_180, clk\_270) can be generated with very high accuracy and low jitter across a wide range of frequencies.

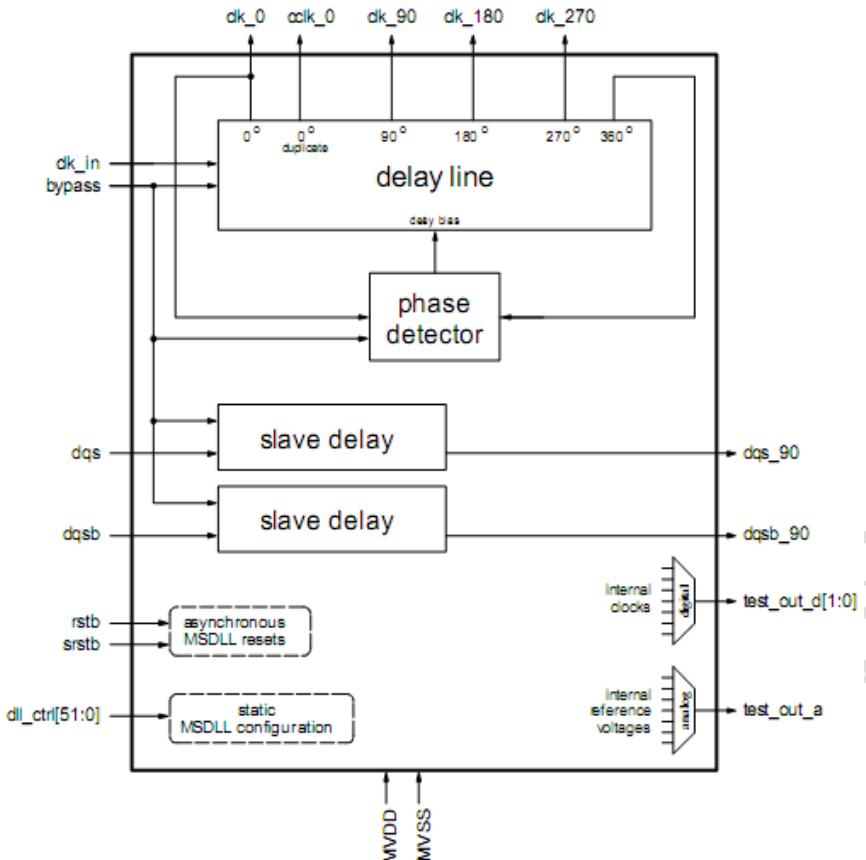


Fig. 13-6 DDR PHY master-slave DLL architecture diagram

The slave delay pair uses timing reference from the delay line to provide a highly accurate 90° delay to `dqs` and `dqsb` inputs (generating `dqs_90` and `dqsb_90` respectively).

A number of test modes and configuration settings are included:

- A bypass mode shuts down all analog circuitry, and directly buffers the input clock and strobes with appropriate delays and inversions to the output clocks and strobes. This mode can be used for low speed functional or IDDQ testing.
- A digital test output (`test_out_d`) provides direct observability of several internal reference clock and timing nodes.
- An analog test output (`test_out_a`) provides direct observability of several internal reference voltages.

The primary application for MSDLL is a DDR2 Byte Lane PHY with Interface Timing Modules (ITMs).

### MSDLL Control for Trim and Test

The performance and testing of the MSDLL can be accessed through the `dll_ctrl` bus. Many of these controls are the same as the MDLL, therefore, this section only describes the settings that are different.

Table 13-9 DDR PHYMSDLL control for trim and test

Static Input	Field	Description
<code>dll_ctrl</code>	[1:0]	Reserved
	[4:2] <code>ipump_trm[2:0]</code>	Charge pump current trim
	[5] <code>test_ctrl_en</code>	Test control enable for analog and digital test outputs
	[8:6] <code>test_ctrl_d[2:0]</code>	Digital test control. Selects the digital signal to be viewed at the digital test output

	[10:9]	test_ctrl_a[1:0]	Analog test control. Selects the analog signal to be viewed at the analog test output
	[11]	test_ctrl_switch	Test control switch. Selects the analog and digital test signals of master or slave
	[14:12]	bias_trm[2:0]	Master bias generator frequency trim
	[19,15]	fdtrm[1:0]	Master bypass fixed delay trim
	[18:16]	bias_trm[6:4]	Master bias generator control voltage trim
	[22:20]	sl_bias_trm[2:0]	Slavebias generator control voltage trim
	[23]	bps200	Bypass frequency select
	[26:24]	sl_bias_trm[6:4]	Slave bias generator control voltage trim
	[28:27]	fdtrm_sl[1:0]	Slave bypass fixed delay trim
	[29]	lock_det_en	Lock detector enable
	[31:30]		Reserved
	[37:32]	sl_fb_trm[5:0]	Slave feedback delay adjust
	[43:38]	fb_trm[5:0]	Master feedback delay adjust
	[45:44]	sl_bypass_start_up[1:0]	Slave auto-startup bypass
	[49:46]	sl_phase_trm[3:0]	Slave phase lock trim
	[50]	test_hizb_a	Analog test output tri-stated control
	[51]		Reserved

## MSDLL Digital Test Control:

Table 13-10 MSDLL digital test control in dll\_ctrl

test_ctrl_en	test_ctrl_switch	test_ctrl_d[2:0]	Function	Suggested Default
0	x	xx	digital test outputs disabled (drive '0')	
1		000	0°output clock (clk_0)	
1		001	90°output clock (clk_90)	
1		010	180° output clock (clk_180)	
1		011	270° output clock (clk_270)	
1		100	360° internal clock (clk_360_int)	
1		101	Master speed-up pulse (spdup)	
1		110	Master slow-down pulse (slwdn)	
1		111	Output clock (cclk_0)	
1		000	Input signal dqs	
1		001	Slave input clock reference (clk_90_in)	
1		010	Slave internal feedback clock (clk_0_out)	
1		011	Output signal dqsb_90	
1		100	Output signal dqs_90	
1		101	Slave speed-up pulse (spdup)	
1		110	Slave slow-down pulse (slwdn)	
1		111	Auto-lock enable signal	0,0,000

## MSDLL Analog Test Control:

Table 13-11 MSDLL analog test control in dll\_ctrl

test_hizb_a	test_ctrl_en	test_ctrl_switch	test_ctrl_a[1:0]	Function	Suggested Default

0	x	x	xx	Tri-state	0,0,0,00
1	0	x	xx	MVSS	
1	1	0	00	Master Filter output (Vc)	
1	1		01	Master Replica bias output for NMOS (Vbn)	
1	1		10	Master Replica bias output for PMOS (Vbp)	
1	1		11	MVDD	
1	1		00	Slave Filter output (Vc)	
1	1	1	01	Slave Replica bias output for NMOS (Vbn)	
1	1		10	Slave Replica bias output for PMOS (Vbp)	
1	1		11	MVDD	

#### MSDLL Lock Detector Enable:

This setting enables start of the slave DLL section after the master DLL section has reached lock. Characteristics of the DLL that may warrant an adjustment of this trim value include the slave DLL delay remaining in it's reset state (minimum delay, much less than 90 degrees) after the DLL lock time.

Table 13-12 MSDLL lock detector enable in dll\_ctrl

Field	Setting	Function	Suggested Default
lock_det_en	0	Disable lock detector	0
	1	Enable lock detector	

#### Slave Auto-Startup Bypass:

By default, the slave DLL automatically starts to lock during the time the master is locking, after the master has begun to approach lock. This setting permits the user to manually start-up the slave DLL. To bypass the automatic startup, this setting should be set to '10'. Once the specified number of clocks has passed for the master DLL to achieve lock, the user sets this field to '11' to permit the slave DLL to startup. The user then waits for the specified number of clocks for the slave DLL to lock before proceeding. Characteristics of the slave DLL that might warrant a manual startup of the slave DLL include the inability for the slave DLL to produce a consistent and/or correct phase difference between the input signal and the output signal.

Table 13-13 slave auto\_startup bypass in dll\_ctrl

sl_bypass_start_up[1:0]	Function	Suggested Default
0X	Slave DLL automatically starts up	00
10	Slave DLL's automatic startup is disabled; the phase detector is disabled	
11	Slave DLL's automatic startup is disabled; the phase detector is enabled	

#### Slave DLL Phase Trim:

Selects the phase difference between the input signal and the corresponding output signal of the slave DLL. This setting applies to the dqs to dqs\_90 and dqsb to dqsb\_90 paths. The nominal phase difference is 90 degrees. Users may select to modify this value to account for factors external to the DLL, which require the DLL to produce a delay of greater than or less than the nominal 90 degrees. When modifying the value of these bits, the user does not need to issue a reset to the DLL but should wait the equivalent of the DLL lock time before the slave DLL circuitry is used (such as, receiving Read data from an SDRAM) to ensure the DLL has adequate time to stabilize with the new settings.

Table 13-14 slave DLL phase trim in dll\_ctrl

sl_phase_trm[3:0]	Phase Difference (degrees)	Suggested Default
0000	90	
0001	72	
0010	54	
0011	36	
0100	108	
0101	90	
0110	72	
0111	54	
1000	126	
1001	108	
1010	90	
1011	72	
1100	144	
1101	126	
1110	108	
1111	90	

0000

### MSDLL Bypass Mode

The DLL bypass mode, when enabled, shuts down all analog delay paths and phase detection circuitry and generates output clocks as directly buffered and inverted versions of clk\_in. Bypass mode can be used for low-speed functional testing or for IDDQ testing. Bypass mode can also be used when operating with LPDDR SDRAMs. When bypass mode is enabled, all analog circuitry is disabled, and all static current paths are shut down. Phased outputs are generated during bypass with inverters and standard delays:

```

clk_0      = buffered clk_in
clk_90     = delayed version of clk_0
clk_180    = inverted clk_0
clk_270    = inverted clk_90
cclk_0     = buffered clk_in
dqs_90     = delayed version of dqs
dqsb_90   = delayed version of dqsb

```

Bypass mode has two settings for the clk\_90 delay to optimize it for two different frequency ranges same as MDLL.

It is also possible to trim the MDLL 90 degree delay using the fdtrm control bits same as MDLL. And it is also possible to trim the MSDLL 90 degree delay using the fdtrm\_sl control bits same as fdtrm.

### 13.4.5 DQS Gating

DDR2 systems use a bidirectional data strobe which is driven by the host during memory writes, and by the SDRAM during memory reads. During active read commands, the ITMS basically acts as a buffer for the incoming DQS/DQS\_b. A turn-around time exists between operations when neither device is driving the bus, and the strobe traces are held by termination circuitry at a mid-rail voltage.

While the DQS lines are held at mid-rail during inactive periods, an unknown value X is being received by the SSTL inputs. To prevent X from causing false transitions and other negative effects within the read path, the input read dqs strobe path is disabled when there is no active read data. The ITMS provides the functions to enable/disable this path, while the control of

these functions is provided by the memory controller logic. A basic view of the enable/disable requirements is shown in following figure.

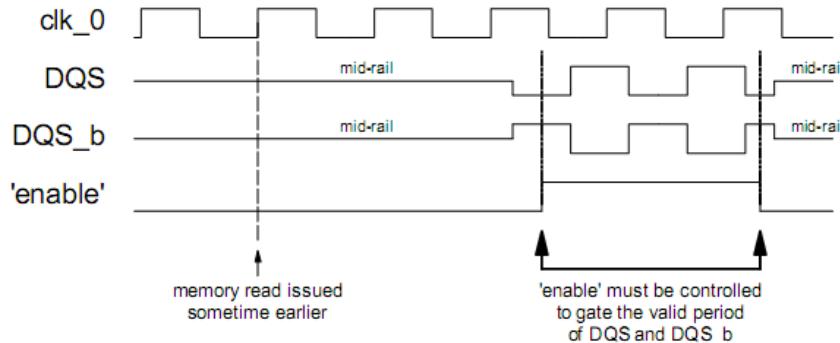


Fig. 13-7 Strobe Gating Requirements During Read Operations

After a read is issued, the SDRAM drives DQS and DQS\_b for a number of clock cycles equal to the read burst length. Differing SDRAM CAS latencies, clock cycle times, board trace lengths, and other analog factors between controller and SDRAM result in a variable latency between when the read was issued, and when the returning DQS/DQS\_b strobes reach the ITMS. The goal of DQS gating is to control a window, which enables and disables the input read dqs path only when the DQS lines are active, not when they are at mid-rail. There is a pre-preamble and post-preamble surrounding the active DQS edges that is used as the point to perform the enabling and disabling of this window.

There are two windowing schemes supported by the ITMS - passive windowing and active windowing - which are selected by input dqs\_config.

### Passive Windowing

In the passive windowing mode (dqs\_config = 1), the controller asserts dqs\_en at the start of the window and de-asserts dqs\_en at the end of the window. This provides the coarse (clock-cycle) position of the enable and disable edges. Fine tuning (1/4 clock cycle) of the window placement is selected by phase\_sel[1:0].

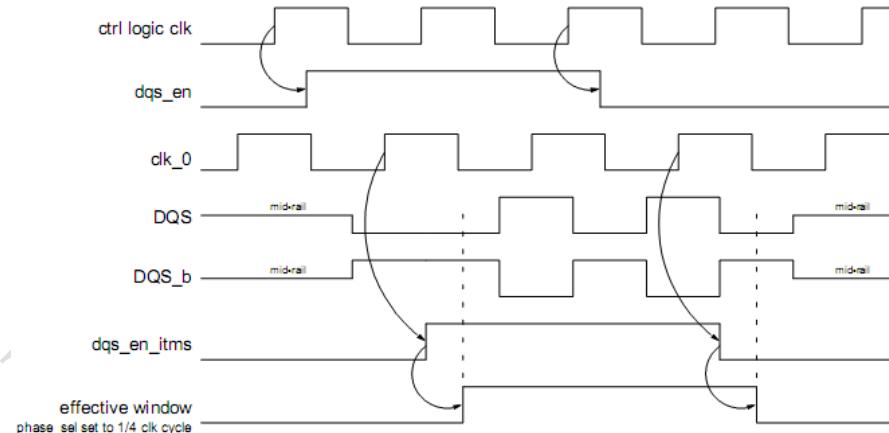


Fig. 13-8 DQS gating – passive windowing mode

The phase\_sel[1:0] settings are provided.

Table 13-15 phase selection for dqs gating

phase_sel[1:0] Phase Selection		
Setting	Selected Phase	Offset
00	clk_90 (90 deg)	1/4 clock cycle
01	clk_180 (180 deg)	1/2 clock cycle
10	clk_270 (270 deg)	3/4 clock cycle
11	clk_0 (360 deg)	1 clock cycle

## Active Windowing

The active windowing mode addresses the fact that the postamble is shorter than the preamble. The optimal window position for the preamble and postamble are not necessarily the same. In the active windowing mode (`dqs_config = 0`), the controller asserts `dqs_en` for one clock cycle at the start of the window and asserts `dqs_dis` for one clock cycle at the end of the window. Internal to ITMS, the assertion of `dqs_dis` is shifted by a further 180 degrees to account for the fact that `DQS_b` occurs 180 degrees later than `DQS`. This provides the coarse (clock-cycle) position of the enable and disable edges.

Fine tuning (1/4 clock cycle) of the window placement is selected by `phase_sel[1:0]`. The effective window is opened in the same manner as in the passive windowing mode, such as `dqs_en` assertion plus the `phase_sel` offset. To close the window, the controller asserts `dqs_dis` to inform the ITMS to expect the last `DQS_b` rising edge of the burst. The `phase_sel` setting is applied to this to set the effective time at which to expect the last `DQS_b` rising edge. The last `DQS_b` rising edge of the burst is also the last data of the burst. This last `DQS_b` rising edge is used to close the window. Thus, the window is self-closing.

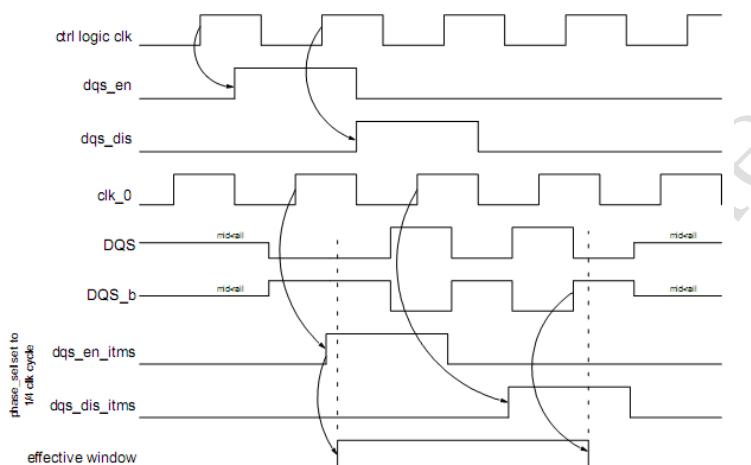


Fig. 13-9 DQS gating – active windowing mode

### 13.4.6 Dynamic Strobe Drift Detection

DDR2 systems can have a long round-trip path from the controller clock output (CK), to the SDRAM, and back to the controller data strobe input (DQS). The sum of potential variations in this path can exceed 25% of a clock cycle at high frequencies (>300MHz), so some compensation should be made if the path delay increases or decreases slowly, but significantly, during normal operation.

The ITMS component has a two-bit strobe drift indicator (`dqs_drift`), which changes value in grey code if the returning strobe drifts across internal 90° timing reference boundaries. The absolute value of this indicator is not important, but the change in value over time is.

Table 13-16 dynamic strobe drift indicators

<code>dqs_drift[1:0]</code>		DQS Drift Direction	Required Changes
Old Value	New Value		
00	01	forward	increase read data latency by 90 degrees
	10	backward	decrease read data latency by 90 degrees
01	11	forward	increase read data latency by 90 degrees
	00	backward	decrease read data latency by 90 degrees
10	00	forward	increase read data latency by 90 degrees
	11	backward	decrease read data latency by 90 degrees
11	10	forward	increase read data latency by 90 degrees
	01	backward	decrease read data latency by 90 degrees

## 13.5 Register description

### 13.5.1 Registers Summary

Name	Offset	Size	Reset Value	Description
DDR_PCTL_SCFG	0x0000	W	0x00000300	State Configuration Register
DDR_PCTL_SCTL	0x0004	W	0x00000000	Operational State Control Register
DDR_PCTL_STAT	0x0008	W	0x00000000	Operational State Status Register
DDR_PCTL_INTRSTAT	0x000c	W	0x00000000	Interrupt Status Register
DDR_PCTL_MCMD	0x0040	W	0x00100000	Memory Command Register
DDR_PCTL_POWCTL	0x0044	W	0x00000000	Power Up Control Register
DDR_PCTL_POWSTAT	0x0048	W	0x00000000	Power Up Status Register
DDR_PCTL_CMDTSTAT	0x004c	W	0x00000000	Command Timers Status Register
DDR_PCTL_CMDTSTATEN	0x0050	W	0x00000000	Command Timers Status Enable Register
DDR_PCTL_MRRCFG0	0x0060	W	0x00000000	Mode Register Read Configuration 0
DDR_PCTL_MRRSTAT0	0x0064	W	0x00000000	Mode Register Read Status 0 Register
DDR_PCTL_MRRSTAT1	0x0068	W	0x00000000	Mode Register Read Status 0 Register
DDR_PCTL_MCFG	0x0080	W	0x00040020	Memory Configuration Register
DDR_PCTL_PPFCFG	0x0084	W	0x00000000	Partially Populated Memories Configuration Register
DDR_PCTL_MSTAT	0x0088	W	0x00000000	Memory Status Register
DDR_PCTL_LPDDR2ZQCFG	0x008c	W	0xab0a560a	LPDDR2 ZQ Configuration Register
DDR_PCTL_MCFG1	0x007c	W	0x00000000	Memory Configuration 1 Register
DDR_PCTL_DTUPDES	0x0094	W	0x00000000	DTU Status Register
DDR_PCTL_DTUNA	0x0098	W	0x00000000	DTU Number of Addresses Created Register
DDR_PCTL_DTUNE	0x009c	W	0x00000000	DTU Number of Errors Register
DDR_PCTL_DTUPRD0	0x00a0	W	0x00000000	DTU Parallel Read 0 Register
DDR_PCTL_DTUPRD1	0x00a4	W	0x00000000	DTU Parallel Read 1 Register
DDR_PCTL_DTUPRD2	0x00a8	W	0x00000000	DTU Parallel Read 2 Register
DDR_PCTL_DTUPRD3	0x00ac	W	0x00000000	DTU Parallel Read 3 Register
DDR_PCTL_DTUAWDT	0x00b0	W	0x00000290	DTU Address Width Register
DDR_PCTL_TOGCNT1U	0x00c0	W	0x00000064	Toggle Counter 1us Register
DDR_PCTL_TINIT	0x00c4	W	0x000000c8	t_init Timing Register

Name	Offset	Size	Reset Value	Description
DDR_PCTL_TRSTH	0x00c8	W	0x00000000	t_rsth Timing Register
DDR_PCTL_TOGCNT100N	0x00cc	W	0x00000001	Toggle Counter 100ns
DDR_PCTL_TREFI	0x00d0	W	0x00000001	t_refi Timing Register
DDR_PCTL_TMRD	0x00d4	W	0x00000001	t_mrd Timing Register
DDR_PCTL_TRFC	0x00d8	W	0x00000001	t_rfc Timing Register
DDR_PCTL_TRP	0x00dc	W	0x00010006	t_trp Timing Register
DDR_PCTL_TRTW	0x00e0	W	0x00000002	t_rtw Timing Register
DDR_PCTL_TAL	0x00e4	W	0x00000000	AL Register
DDR_PCTL_TCL	0x00e8	W	0x00000004	CL Timing Register
DDR_PCTL_TCWL	0x00ec	W	0x00000003	CWL Timing Register
DDR_PCTL_TRAS	0x00f0	W	0x00000010	t_ras Timing Register
DDR_PCTL_TRC	0x00f4	W	0x00000016	t_rc Timing Register
DDR_PCTL_TRCD	0x00f8	W	0x00000006	t_rcd Timing Register
DDR_PCTL_TRRD	0x00fc	W	0x00000004	t_rrd Timing Register
DDR_PCTL_TRTP	0x0100	W	0x00000003	t_rtp Timing Register
DDR_PCTL_TWR	0x0104	W	0x00000006	t_wr Register
DDR_PCTL_TWTR	0x0108	W	0x00000004	t_wtr Timing Register
DDR_PCTL_TEXSR	0x010c	W	0x00000001	t_exsr Timing Register
DDR_PCTL_TXP	0x0110	W	0x00000001	t_xp Timing Register
DDR_PCTL_TXPDLL	0x0114	W	0x00000000	t_xpdll Timing Register
DDR_PCTL_TZQCS	0x0118	W	0x00000000	t_zqcs Timing Register
DDR_PCTL_TZQCSI	0x011c	W	0x00000000	t_zqcsi Timing Register
DDR_PCTL_TDQS	0x0120	W	0x00000001	t_dqs Timing Register
DDR_PCTL_TCKSRE	0x0124	W	0x00000000	t_cksre Timing Register
DDR_PCTL_TCKSRX	0x0128	W	0x00000000	t_cksrx Timing Register
DDR_PCTL_TCKE	0x012c	W	0x00000003	t_cke Timing Register
DDR_PCTL_TMOD	0x0130	W	0x00000000	t_mod Timing Register
DDR_PCTL_TRSTL	0x0134	W	0x00000000	Reset Low Timing Register
DDR_PCTL_TZQCL	0x0138	W	0x00000000	t_zqcl Timing Register
DDR_PCTL_TMRR	0x013c	W	0x00000002	t_mrr Timing Register
DDR_PCTL_TCKESR	0x0140	W	0x00000004	t_ckesr Timing Register
DDR_PCTL_TDPD	0x0144	W	0x00000000	t_dpd Timing Register
DDR_PCTL_DTUWACTL	0x0200	W	0x00000000	DTU Write Address Control
DDR_PCTL_DTURACTL	0x0204	W	0x00000000	DTU Read Address Control Register
DDR_PCTL_DTUCFG	0x0208	W	0x00000000	DTU Configuration Control Register
DDR_PCTL_DTUECTL	0x020c	W	0x00000000	DTU Execute Control Register
DDR_PCTL_DTUWD0	0x0210	W	0x00000000	DTU Write Data #0 Register
DDR_PCTL_DTUWD1	0x0214	W	0x00000000	DTU Write Data #1 Register
DDR_PCTL_DTUWD2	0x0218	W	0x00000000	DTU Write Data #2 Register
DDR_PCTL_DTUWD3	0x021c	W	0x00000000	DTU Write Data #3 Register
DDR_PCTL_DTUWDM	0x0220	W	0x00000000	DTU Write Data Mask Register

Name	Offset	Size	Reset Value	Description
DDR_PCTL_DTURD0	0x0224	W	0x00000000	DTU Read Data #0 Register
DDR_PCTL_DTURD1	0x0228	W	0x00000000	DTU Read Data #1 Register
DDR_PCTL_DTURD2	0x022c	W	0x00000000	DTU Read Data #2 Register
DDR_PCTL_DTURD3	0x0230	W	0x00000000	DTU Read Data #3 Register
DDR_PCTL_DTULFSRWD	0x0234	W	0x00000000	DTU LFSR Seed for Write Data Generation Register
DDR_PCTL_DTULFSRRD	0x0238	W	0x00000000	DTU LFSR Seed for Read Data Generation Register
DDR_PCTL_DTUEAF	0x023c	W	0x00000000	DTU Error Address FIFO Register
DDR_PCTL_DFITCTRLDELAY	0x0240	W	0x00000002	DFI tctrl_delay Register
DDR_PCTL_DFIODTCFG	0x0244	W	0x00000000	DFI ODT Configuration
DDR_PCTL_DFIODTCFG1	0x0248	W	0x06060000	DFI ODT Timing Configuration 1 (for Latency and Length)
DDR_PCTL_DFIODTRANKMAP	0x024c	W	0x00008421	DFI ODT Rank Mapping
DDR_PCTL_DFITPHYWRDATA	0x0250	W	0x00000001	DFI tphy_wrdata Register
DDR_PCTL_DFITPHYWRLAT	0x0254	W	0x00000001	DFI tphy_wrlat Register
DDR_PCTL_DFITRDDATAEN	0x0260	W	0x00000001	DFI trddata_en Register
DDR_PCTL_DFITPHYRDLAT	0x0264	W	0x0000000f	DFI tphy_rdlat Register
DDR_PCTL_DFITPHYUPDTYPE0	0x0270	W	0x00000010	DFI tphyupd_type0 Register
DDR_PCTL_DFITPHYUPDTYPE1	0x0274	W	0x00000010	DFI tphyupd_type1 Register
DDR_PCTL_DFITPHYUPDTYPE2	0x0278	W	0x00000010	DFI tphyupd_type2 Register
DDR_PCTL_DFITPHYUPDTYPE3	0x027c	W	0x00000010	DFI tphyupd_type3 Register
DDR_PCTL_DFITCTRLUPDMIN	0x0280	W	0x00000010	DFI tctrlupd_min Register
DDR_PCTL_DFITCTRLUPDMAX	0x0284	W	0x00000040	DFI tctrlupd_max Register
DDR_PCTL_DFITCTRLUPDDLY	0x0288	W	0x00000008	DFI tctrlupddly Register
DDR_PCTL_DFIUPDCFG	0x0290	W	0x00000003	DFI Update Configuration Register
DDR_PCTL_DFITREFMSKI	0x0294	W	0x00000000	DFI Masked Refresh Interval
DDR_PCTL_DFITCTRLUPDI	0x0298	W	0x00000000	DFI tctrlupd_interval Register
DDR_PCTL_DFITRCFG0	0x02ac	W	0x00000000	DFI Training Configuration 0 Register
DDR_PCTL_DFITRSTAT0	0x02b0	W	0x00000000	DFI Training Status 0 Register

Name	Offset	Size	Reset Value	Description
DDR_PCTL_DFITRWRLVLEN	0x02b4	W	0x00000000	DFI Training dfi_wrlvl_en Register
DDR_PCTL_DFITRRDLVLEN	0x02b8	W	0x00000000	DFI Training dfi_rdlvl_en Register
DDR_PCTL_DFITRRDLVLGATEEN	0x02bc	W	0x00000000	DFI Training dfi_rdlvl_gate_en Register
DDR_PCTL_DFISTSTAT0	0x02c0	W	0x00000000	DFI Status Status 0 Register
DDR_PCTL_DFISTCFG0	0x02c4	W	0x00000000	DFI Status Configuration 0 Register
DDR_PCTL_DFISTCFG1	0x02c8	W	0x00000000	DFI Status Configuration 1 Register
DDR_PCTL_DFITDRAMCLKEN	0x02d0	W	0x00000002	DFI tdrum_clk_enable Register
DDR_PCTL_DFITDRAMCLKDIS	0x02d4	W	0x00000002	DFI tdrum_clk_disable Register
DDR_PCTL_DFISTCFG2	0x02d8	W	0x00000000	DFI Status Configuration 2 Register
DDR_PCTL_DFISTPARCLR	0x02dc	W	0x00000000	DFI Status Parity Clear Register
DDR_PCTL_DFISTPARLOG	0x02e0	W	0x00000000	DFI Status Parity Log Register
DDR_PCTL_DFILPCFG0	0x02f0	W	0x00070000	DFI Low Power Configuration 0 Register
DDR_PCTL_DFITRWRLVLRESP0	0x0300	W	0x00000000	DFI Training dfi_wrlvl_resp Status 0 Register
DDR_PCTL_DFITRWRLVLRESP1	0x0304	W	0x00000000	DFI Training dfi_wrlvl_resp Status 1 Register
DDR_PCTL_DFITRWRLVLRESP2	0x0308	W	0x00000000	DFI Training dfi_wrlvl_resp Status 2 Register
DDR_PCTL_DFITRRDLVLRESP0	0x030c	W	0x00000000	DFI Training dfi_rdlvl_resp Status 0 Register
DDR_PCTL_DFITRRDLVLRESP1	0x0310	W	0x00000000	DFI Training dfi_rdlvl_resp Status 1 Register
DDR_PCTL_DFITRRDLVLRESP2	0x0314	W	0x00000000	DFI Training dfi_rdlvl_resp Status 2 Register
DDR_PCTL_DFITRWRLVLDelay0	0x0318	W	0x00000000	DFI Training dfi_wrlvl_delay Configuration 0 Register
DDR_PCTL_DFITRWRLVLDelay1	0x031c	W	0x00000000	DFI Training dfi_wrlvl_delay Configuration 1 Register
DDR_PCTL_DFITRWRLVLDelay2	0x0320	W	0x00000000	DFI Training dfi_wrlvl_delay Configuration 2 Register
DDR_PCTL_DFITRRDLVLDelay0	0x0324	W	0x00000000	DFI Training dfi_rdlvl_delay Configuration 0 Register
DDR_PCTL_DFITRRDLVLDelay1	0x0328	W	0x00000000	DFI Training dfi_rdlvl_delay Configuration 1 Register

Name	Offset	Size	Reset Value	Description
DDR_PCTL_DFITRRDLVLDELAY2	0x032c	W	0x00000000	DFI Training dfi_rdlvl_delay Configuration 2 Register
DDR_PCTL_DFITRRDLVLGATEDELAY0	0x0330	W	0x00000000	DFI Training dfi_rdlvl_gate_delay Configuration 0
DDR_PCTL_DFITRRDLVLGATEDELAY1	0x0334	W	0x00000000	DFI Training dfi_rdlvl_gate_delay Configuration 1
DDR_PCTL_DFITRRDLVLGATEDELAY2	0x0338	W	0x00000000	DFI Training dfi_rdlvl_gate_delay Configuration 2
DDR_PCTL_DFITRCMD	0x033c	W	0x00000000	DFI Training Command Register
DDR_PCTL_IPVR	0x03f8	W	0x00000000	IP Version Register
DDR_PCTL_IPTR	0x03fc	W	0x44574300	IP Type Register

Name	Offset	Size	Reset Value	Description
DDR_PUBL_RIDR	0x0000	W	0x00100140	Revision Identification Register
DDR_PUBL_PIR	0x0004	W	0x00000000	PHY Initialization Register
DDR_PUBL_PGCR	0x0008	W	0x01bc2e04	PHY General Configuration Register
DDR_PUBL_PGSR	0x000c	W	0x00000000	PHY General Status Register
DDR_PUBL_DLLGCR	0x0010	W	0x03737000	DLL General Control Register
DDR_PUBL_ACDLLCR	0x0014	W	0x40000000	AC DLL Control Register
DDR_PUBL_PTR0	0x0018	W	0x0022af9b	PHY Timing Register 0
DDR_PUBL_PTR1	0x001c	W	0x0604111d	PHY Timing Register 1
DDR_PUBL_PTR2	0x0020	W	0x042da072	PHY Timing Register 2
DDR_PUBL_ACIOCR	0x0024	W	0x33c03812	AC I/O Configuration Register
DDR_PUBL_DXCCR	0x0028	W	0x00000800	DATX8 Common Configuration Register
DDR_PUBL_DSGCR	0x002c	W	0xfa00001f	DDR System General Configuration Register
DDR_PUBL_DCR	0x0030	W	0x0000000b	DRAM Configuration Register
DDR_PUBL_DTPR0	0x0034	W	0x3092666e	DRAM Timing Parameters Register 0
DDR_PUBL_DTPR1	0x0038	W	0x09830090	DRAM Timing Parameters Register 1
DDR_PUBL_DTPR2	0x003c	W	0x1001a0c8	DRAM Timing Parameters Register 2
DDR_PUBL_MR0	0x0040	W	0x00000a52	Mode Register 0
DDR_PUBL_MR1	0x0044	W	0x00000000	Mode Register 1
DDR_PUBL_MR2	0x0048	W	0x00000000	Mode Register 2
DDR_PUBL_MR3	0x004c	W	0x00000000	Mode Register 3

Name	Offset	Size	Reset Value	Description
DDR_PUBL_ODTCSR	0x0050	W	0x00210000	ODT Configuration Register
DDR_PUBL_DTAR	0x0054	W	0x00000000	Data Training Address Register
DDR_PUBL_DTDRO	0x0058	W	0xdd22ee11	Data Training Data Register 0
DDR_PUBL_DTDRI	0x005c	W	0x7788bb44	Data Training Data Register 1
DDR_PUBL_DCUAR	0x00c0	W	0x00000000	DCU Address Register
DDR_PUBL_DCUDR	0x00c4	W	0x00000000	DCU Data Register
DDR_PUBL_DCURR	0x00c8	W	0x00000000	DCU Run Register
DDR_PUBL_DCULR	0x00cc	W	0x00000000	DCU Loop Register
DDR_PUBL_DCUGCR	0x00d0	W	0x00000000	DCU General Configuration Register
DDR_PUBL_DCUTPR	0x00d4	W	0x00000000	DCU Timing Parameters Registers
DDR_PUBL_DCUSR0	0x00d8	W	0x00000000	DCU Status Register 0
DDR_PUBL_DCUSR1	0x00dc	W	0x00000000	DCU Status Register 1
DDR_PUBL_BISTRR	0x0100	W	0x00000000	BIST Run Register
DDR_PUBL_BISTMSKR0	0x0104	W	0x00000000	BIST Mask Register 0
DDR_PUBL_BISTMSKR1	0x0108	W	0x00000000	BIST Mask Register 1
DDR_PUBL_BISTWCR	0x010c	W	0x00000020	BIST Word Count Register
DDR_PUBL_BISTLSR	0x0110	W	0x1234abcd	BIST LFSR Seed Register
DDR_PUBL_BISTAR0	0x0114	W	0x00000000	BIST Address Register 0
DDR_PUBL_BISTAR1	0x0118	W	0x0000000c	BIST Address Register 1
DDR_PUBL_BISTAR2	0x011c	W	0x7fffffff	BIST Address Register 2
DDR_PUBL_BISTUDPR	0x0120	W	0xfffff000	BIST User Data Pattern Register
DDR_PUBL_BISTGSR	0x0124	W	0x00000000	BIST General Status Register
DDR_PUBL_BISTWER	0x0128	W	0x00000000	BIST Word Error Register
DDR_PUBL_BISTBER0	0x012c	W	0x00000000	BIST Bit Error Register 0
DDR_PUBL_BISTBER1	0x0130	W	0x00000000	BIST Bit Error Register 1
DDR_PUBL_BISTBER2	0x0134	W	0x00000000	BIST Bit Error Register 2
DDR_PUBL_BISTWCSR	0x0138	W	0x00000000	BIST Word Count Status Register
DDR_PUBL_BISTFWR0	0x013c	W	0x00000000	BIST Fail Word Register 0
DDR_PUBL_BISTFWR1	0x0140	W	0x00000000	BIST Fail Word Register 1
DDR_PUBL_ZQ0CR0	0x0180	W	0x0000014a	ZQ 0 Impedance Control Register 0
DDR_PUBL_ZQ0CR1	0x0184	W	0x0000007b	ZQ 0 Impedance Control Register 1
DDR_PUBL_ZQ0SR0	0x0188	W	0x00000000	ZQ 0 Impedance Status Register 0
DDR_PUBL_ZQ0SR1	0x018c	W	0x00000000	ZQ 0 Impedance Status Register 1
DDR_PUBL_ZQ1CR0	0x0190	W	0x0000014a	ZQ 1 Impedance Control Register 0
DDR_PUBL_ZQ1CR1	0x0194	W	0x0000007b	ZQ 1 Impedance Control Register 1
DDR_PUBL_ZQ1SR0	0x0198	W	0x00000000	ZQ 1 Impedance Status Register 0
DDR_PUBL_ZQ1SR1	0x019c	W	0x00000000	ZQ 1 Impedance Status Register 1

Name	Offset	Size	Reset Value	Description
DDR_PUBL_ZQ2CR0	0x01a0	W	0x00000014a	ZQ 2 Impedance Control Register 0
DDR_PUBL_ZQ2CR1	0x01a4	W	0x00000007b	ZQ 2 Impedance Control Register 1
DDR_PUBL_ZQ2SR0	0x01a8	W	0x000000000	ZQ 2 Impedance Status Register 0
DDR_PUBL_ZQ2SR1	0x01ac	W	0x000000000	ZQ 2 Impedance Status Register 1
DDR_PUBL_ZQ3CR0	0x01b0	W	0x00000014a	ZQ 3 Impedance Control Register 0
DDR_PUBL_ZQ3CR1	0x01b4	W	0x00000007b	ZQ 3 Impedance Control Register 1
DDR_PUBL_ZQ3SR0	0x01b8	W	0x000000000	ZQ 3 Impedance Status Register 0
DDR_PUBL_ZQ3SR1	0x01bc	W	0x000000000	ZQ 3 Impedance Status Register 1
DDR_PUBL_DX0GCR	0x01c0	W	0x000000681	DATX8 0 General Configuration Register
DDR_PUBL_DX0GSR0	0x01c4	W	0x000000000	DATX8 0 General Status Register 0
DDR_PUBL_DX0GSR1	0x01c8	W	0x000000000	DATX8 0 General Status Register 1
DDR_PUBL_DX0DLLCR	0x01cc	W	0x400000000	DATX8 0 DLL Control Register
DDR_PUBL_DX0DQTR	0x01d0	W	0xffffffff	DATX8 0 DQ Timing Register
DDR_PUBL_DX0DQSTR	0x01d4	W	0x3db05000	DATX8 0 DQS Timing Register
DDR_PUBL_DX1GCR	0x0200	W	0x000000681	DATX8 1 General Configuration Register
DDR_PUBL_DX1GSR0	0x0204	W	0x000000000	DATX8 1 General Status Register 0
DDR_PUBL_DX1GSR1	0x0208	W	0x000000000	DATX8 1 General Status Register 1
DDR_PUBL_DX1DLLCR	0x020c	W	0x400000000	DATX8 1 DLL Control Register
DDR_PUBL_DX1DQTR	0x0210	W	0xffffffff	DATX8 1 DQ Timing Register
DDR_PUBL_DX1DQSTR	0x0214	W	0x3db05000	DATX8 1 DQS Timing Register
DDR_PUBL_DX2GCR	0x0240	W	0x000000681	DATX8 2 General Configuration Register
DDR_PUBL_DX2GSR0	0x0244	W	0x000000000	DATX8 2 General Status Register 0
DDR_PUBL_DX2GSR1	0x0248	W	0x000000000	DATX8 2 General Status Register 1
DDR_PUBL_DX2DLLCR	0x024c	W	0x400000000	DATX8 2 DLL Control Register
DDR_PUBL_DX2DQTR	0x0250	W	0xffffffff	DATX8 2 DQ Timing Register
DDR_PUBL_DX2DQSTR	0x0254	W	0x3db05000	DATX8 2 DQS Timing Register
DDR_PUBL_DX3GCR	0x0280	W	0x000000681	DATX8 3 General Configuration Register
DDR_PUBL_DX3GSR0	0x0284	W	0x000000000	DATX8 3 General Status Register 0

Name	Offset	Size	Reset Value	Description
DDR_PUBL_DX3GSR1	0x0288	W	0x00000000	DATX8 3 General Status Register 1
DDR_PUBL_DX3DLLCR	0x028c	W	0x40000000	DATX8 3 DLL Control Register
DDR_PUBL_DX3DQTR	0x0290	W	0xffffffff	DATX8 3 DQ Timing Register
DDR_PUBL_DX3DQSTR	0x0294	W	0x3db05000	DATX8 3 DQS Timing Register

Notes: Size: **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** -WORD (32 bits) access

### 13.5.2 Detail Registers Description

#### DDR\_PCTL\_SCFG

Address: Operational Base + offset (0x0000)

State Configuration Register

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:8	RW	0x3	<p>bbflags_timing The n_bbflags is a NIF output vector which provides combined information about the status of each memory bank. The de-assertion is based on when precharge, activates, reads/writes are scheduled by the TCU block. It may be possible to de-assert n_bbflags earlier than calculated by the TCU block. Programming bbflags_timing is used to achieve this. The maximum recommended value is: UPCTL_TCU_SED_P - TRP.t_rp. The programmed value is the maximum number of "early" cycles that n_bbflags maybe de-asserted. The actual achieved de-assertion depends on the traffic profile. In 1:2 mode the maximum allowed programmable value is 4'b0111 In 1:1 mode the value can be 4'b1111</p>
7	RO	0x0	reserved
6	RW	0x0	<p>nfifo_nif1_dis For internal use only for NFIFO testing. 1'b0: Only supported setting 1'b1: For internal use only</p>
5:1	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	<p>hw_low_power_en</p> <p>Enables the hardware low-power interface. Allows the system to request via hardware (c_sysreq input) to enter the memories into Self-Refresh.</p> <p>The handshaking between the request and acknowledge hardware low power signals (c_sysreq and c_sysack, respectively) is always performed, but the uPCTL response depends on the value set on this register field and by the value driven on the c_active_in input pin.</p> <p>1'b0: Disabled. Requests are always denied and uPCTL is unaffected by c_sysreq</p> <p>1'b1: Enabled. Requests are accepted or denied, depending on the current operational state of uPCTL and on the value of c_active_in</p>

**DDR\_PCTL\_SCTL**

Address: Operational Base + offset (0x0004)

Operational State Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2:0	RW	0x0	<p>state_cmd</p> <p>Issues an operational state transition request to the uPCTL.</p> <p>3'b000: INIT (move to Init_mem from Config)      3'b001: CFG (move to Config from Init_mem or Access)      3'b010: GO (move to Access from Config)      3'b011: SLEEP (move to Low_power from Access)      3'b100: WAKEUP (move to Access from Low_power)      Others: Reserved</p>

**DDR\_PCTL\_STAT**

Address: Operational Base + offset (0x0008)

Operational State Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>								
6:4	RO	0x0	<p>lp_trig Reports the status of what triggered an entry to Low_power state. Is only set if in Low_power state. The individual bits report the following:</p> <ul style="list-style-type: none"> <li>- lp_trig[2]: Software driven due to SCTL.state_cmd==SLEEP</li> <li>- lp_trig[1]: Hardware driven due to Hardware Low Power Interface</li> <li>- lp_trig[0]: Hardware driven due to Auto Self Refresh (MCFG1.sr_idle&gt;0)</li> </ul> <p><i>Note: if more than one trigger happens at the exact same time, more than one bit of lp_trig may be asserted high.</i></p>								
3	RO	0x0	reserved								
2:0	RO	0x0	<p>ctl_stat Returns the current operational state of the uPCTL.</p> <table> <tr><td>3'b000: Init_mem</td></tr> <tr><td>3'b001: Config</td></tr> <tr><td>3'b010: Config_req</td></tr> <tr><td>3'b011: Access</td></tr> <tr><td>3'b100: Access_req</td></tr> <tr><td>3'b101: Low_power</td></tr> <tr><td>3'b110: Low_power_entry_req</td></tr> <tr><td>3'b111: Low_power_exit_req</td></tr> </table>	3'b000: Init_mem	3'b001: Config	3'b010: Config_req	3'b011: Access	3'b100: Access_req	3'b101: Low_power	3'b110: Low_power_entry_req	3'b111: Low_power_exit_req
3'b000: Init_mem											
3'b001: Config											
3'b010: Config_req											
3'b011: Access											
3'b100: Access_req											
3'b101: Low_power											
3'b110: Low_power_entry_req											
3'b111: Low_power_exit_req											

**DDR\_PCTL\_INTRSTAT**

Address: Operational Base + offset (0x000c)

Interrupt Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>		
31:2	RO	0x0	reserved		
1	RO	0x0	<p>parity_intr Indicates that a DFI parity error has been detected</p> <table> <tr><td>1'b0: No error</td></tr> <tr><td>1'b1: Parity error</td></tr> </table>	1'b0: No error	1'b1: Parity error
1'b0: No error					
1'b1: Parity error					
0	RO	0x0	<p>ecc_intr Indicates that an ECC error has been detected</p> <table> <tr><td>1'b0: No error</td></tr> <tr><td>1'b1: Parity error</td></tr> </table>	1'b0: No error	1'b1: Parity error
1'b0: No error					
1'b1: Parity error					

**DDR\_PCTL\_MCMD**

Address: Operational Base + offset (0x0040)

Memory Command Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RWSC	0x0	<p>start_cmd</p> <p>Start command. When this bit is set to 1, the command operation defined in the cmd_opcode field is started. This bit is automatically cleared by the uPCTL after the command is finished. The application can poll this bit to determine when uPCTL is ready to accept another command. This bit cannot be cleared to 1'b0 by software.</p>
30:28	RO	0x0	reserved
27:24	RW	0x0	<p>cmd_add_del</p> <p>Set the additional delay associated with each command to <math>2^n</math> internal timers clock cycles, where n is the bit field value. If n=0, the delay is 0. Max value is n=10.</p>
23:20	RW	0x1	<p>rank_sel</p> <p>Rank select for the command to be executed.</p> <p>4'b0001: Rank 0 4'b0010: Rank 1 4'b0100: Rank 2 4'b1000: Rank 3 4'b0000: Reserved</p> <p>Multiple 1'b1s in rank_sel mean multiple ranks are selected, which is useful broadcasting commands in parallel to multiple ranks during initialization and configuration of the memories.</p> <p>If MCMD.cmd_opcode=RSTL, all ranks should be selected as it cannot be performed to individual ranks.</p>
19:17	RW	0x0	<p>bank_addr</p> <p>Mode Register address driven on the memory bank address bits, BA1, BA0, during a Mode Register Set operation, defined by cmd_opcode=MRS. For other values of cmd_opcode, this field is ignored.</p> <p>3'b000: MR0 (MR in DDR2) 3'b001: MR1 (EMR in DDR2) 3'b010: MR2 (EMR(2) in DDR2) 3'b011: MR3 (EMR(3) in DDR2) Others: Reserved</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16:4	RW	0x0000	cmd_addr Mode Register value driven on the memory address bits, A12 to A0, during a Mode Register Set operation defined by cmd_opcode=MRS. For other values of cmd_opcode this field is ignored. Refer to the memory specification for the correct settings of the various bits of this field during a MRS operation. If LPDDR2, this fields is merged into bank_addr - lpddr2_addr.
3:0	RW	0x0	cmd_opcode Command to be issued to the memory. 4'b0000: Deselect. This is only used for timing purposes, no actual direct Deselect command is passed to the memories 4'b0001: Precharge All (PREA) 4'b0010: Refresh (REF) 4'b0011: Mode Register Set (MRS) - is MRW in LPDDR2, MRS otherwise 4'b0100: ZQ Calibration Short (ZQCS, only applies to LPDDR2/DDR3) 4'b0101: ZQ Calibration Long (ZQCL, only applies to LPDDR2/DDR3) 4'b0110: Software Driven Reset (RSTL, only applies to DDR3) 4'b0111: Reserved 4'b1000: Mode Register Read (MRR) - is MRR in LPDDR2, is SRR in mDRR and is MPR in DDR3 4'b1001: Deep Power Down Entry (DPDE, only applies to mDDR/LPDDR2) Others: Reserved

**DDR\_PCTL\_POWCTL**

Address: Operational Base + offset (0x0044)

Power Up Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RWSC	0x0	power_up_start Start the memory power up sequence. When this bit is set to 1'b1, uPCTL starts the CKE and RESET power up sequence to the memories. This bit is automatically cleared by uPCTL after the sequence is completed. This bit cannot be cleared to 1'b0 by software.

**DDR\_PCTL\_POWSTAT**

Address: Operational Base + offset (0x0048)

Power Up Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RO	0x0	power_up_done Returns the status of the memory power-up sequence. 1'b0: Power-up sequence has not been performed 1'b1: Power-up sequence has been performed

**DDR\_PCTL\_CMDTSTAT**

Address: Operational Base + offset (0x004c)

Command Timers Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RO	0x0	cmd_tstat Returns the status of the timers for memory commands. This ANDs all the command timers together. 1'b0: One or more command timers has not expired 1'b1: All command timers have expired

**DDR\_PCTL\_CMDTSTATEN**

Address: Operational Base + offset (0x0050)

Command Timers Status Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	cmd_tstat_en Enables the generation of the status of the timers for memory commands. Is enabled before CMDTSTAT register is read. 1'b0: Disabled 1'b1: Enabled

**DDR\_PCTL\_MRRCFG0**

Address: Operational Base + offset (0x0060)

Mode Register Read Configuration 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	mrr_byte_sel Selects which byte's data to store when performing an MRR command via MCMD. LegalValues: 0 .. 8

**DDR\_PCTL\_MRRSTAT0**

Address: Operational Base + offset (0x0064)

Mode Register Read Status 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x00	mrrstat_beat3 MRR/MPR read data beat 3
23:16	RO	0x00	mrrstat_beat2 MRR/MPR read data beat 2
15:8	RO	0x00	mrrstat_beat1 MRR/MPR read data beat 1
7:0	RO	0x00	mrrstat_beat0 MRR/MPR read data beat 0

**DDR\_PCTL\_MRRSTAT1**

Address: Operational Base + offset (0x0068)

Mode Register Read Status 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x00	mrrstat_beat7 MRR/MPR read data beat 7
23:16	RO	0x00	mrrstat_beat6 MRR/MPR read data beat 6
15:8	RO	0x00	mrrstat_beat5 MRR/MPR read data beat 5
7:0	RO	0x00	mrrstat_beat4 MRR/MPR read data beat 4

**DDR\_PCTL\_MCFG**

Address: Operational Base + offset (0x0080)

Memory Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	mddr_lpddr2_clock_stop_idle Clock stop idle period in n_clk cycles. Memories are placed into clock stop mode if the NIF is idle for mddr_lpddr2_clkstop_idle n_clk cycles. The automatic clock stop function is disabled when mddr_lpddr2_clkstop_idle=0. Clock stop mode is only applicable in mDDR/LPDDR2.
23:22	RW	0x0	mddr_lpddr2_en mDDR/LPDDR2 Enable. Enables support for mDDR or LPDDR2. 2'b00: mDDR/LPDDR2 Disabled 2'b10: mDDR Enabled 2'b11: LPDDR2 Enabled Others: Reserved.
21:20	RW	0x0	mddr_lpddr2_bl mDDR/LPDDR2 Burst Length. The BL setting must be consistent with the value programmed into the BL field of MR. 2'b00: BL2, Burst length of 2 (MR.BL=3'b001, mDDR only) 2'b01: BL4, Burst length of 4 (MR.BL=3'b010, for mDDR and LPDDR2) 2'b10: BL8, Burst length of 8 (MR.BL=3'b011, for mDDR and LPDDR2) 2'b11: BL16, Burst length of 16 (MR.BL=3'b100, for mDDR and LPDDR2) This value is effective only if MCFG.mddr_lpddr2_en[1]=1'b1. Otherwise, MCFG.mem_bl is used to define uPCTL's Burst Length (for DDR2/DDR3).
19:18	RW	0x1	tfaw_cfg Sets tFAW to be 4, 5 or 6 times tRRD. 2'b00: set tFAW=4*tRRD 2'b01: set tFAW=5*tRRD 2'b10: set tFAW=6*tRRD
17	RW	0x0	pd_exit_mode Selects the mode for Power Down Exit. For DDR2/DDR3, the power down exit mode setting in uPCTL must be consistent with the value programmed into the power down exit mode bit of MRO. For mDDR/LPDDR2, only fast exit mode is valid. 1'b0: slow exit 1'b1: fast exit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16	RW	0x0	pd_type Sets the Power down type. 1'b0: Precharge Power Down 1'b1: Active Power Down
15:8	RW	0x00	pd_idle Power-down idle period in n_clk cycles. Memories are placed into power-down mode if the NIF is idle for pd_idle n_clk cycles. The automatic power down function is disabled when pd_idle=0.
7	RO	0x0	reserved
6	RW	0x0	lpddr2_s4 Enables LPDDR2-S4 support. 1'b0: LPDDR2-S4 disabled (LPDDR2-S2 enabled) 1'b1: LPDDR2-S4 enabled
5	RW	0x1	ddr3_en Select DDR2 or DDR3 protocol. Ignored, if mDDR or LPDDR2 support is enabled. 1'b0: DDR2 Protocol Rules 1'b1: DDR3 Protocol Rules
4	RW	0x0	stagger_cs For multi-rank commands from the DCU, stagger the assertion of CS_N to odd and even ranks by one n_clk cycle. This is useful when using RDIMMs, when multi-rank commands may be interpreted as writes to control words in the register chip. 1'b0: Do not stagger CS_N 1'b1: Stagger CS_N
3	RW	0x0	two_t_en Enables 2T timing for memory commands. 1'b0: Disabled 1'b1: Enabled
2	RW	0x0	bl8int_en Setting this bit enables the BL8 interrupt function of DDR2. This is the capability to early terminate a BL8 after only 4 DDR beats by issuing the next command two cycles earlier. This functionality is only available for DDR2 memories and this setting is ignored for mDDR/LPDDR2 and DDR3. 1'b0: Disabled 1'b1: Enabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	<p>cke_or_en</p> <p>This bit is intended to be set for 4-rank RDIMMs, which have a 2-bit CKE input. If set, dfi_cke[0] is asserted to enable either of the even ranks (0 and 2), while dfi_cke[1] is asserted to enable either of the odd ranks (1 and 3). dfi_cke[3:2] are inactive (0)</p> <p>1'b0: Disabled 1'b1: Enabled</p>
0	RW	0x0	<p>mem_bl</p> <p>DDR Burst Length. The BL setting in DDR2 / DDR3 must be consistent with the value programmed into the BL field of MR0.</p> <p>1'b0: BL4, Burst length of 4 (MR0.BL=3'b010, DDR2 only) 1'b1: BL8, Burst length of 8 (MR0.BL=3'b011 for DDR2, MR0.BL=2'b00 for DDR3)</p>

**DDR\_PCTL\_PPCFG**

Address: Operational Base + offset (0x0084)

Partially Populated Memories Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8:1	RW	0x00	<p>rpmem_dis</p> <p>Reduced Population Disable bits. Setting these bits disables the corresponding NIF/DDR data lanes from writing or reading data. Lane 0 is always present, hence only 8 bits are required for the remaining lanes including the ECC lane.</p> <p>In 1:2 mode bit 0 of rpmem_dis covers n_wdata/n_rdata/m_ctl_d/m_phy_q[63:32], bit 1 [95:64] etc.</p> <p>In 1:1 mode bit 0 of rpmem_dis covers n_wdata/n_rdata/m_ctl_d/m_phy_q[31:16], bit 2 [47:32] etc.</p> <p>There are no restrictions on which byte lanes can be disabled, other than byte lane 0 is required. Gaps between enabled byte lanes are allowed.</p> <p>For each bit:</p> <p>1'b0: lane exists 1'b1: lane is disabled</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	<p>ppmem_en</p> <p>Partially Population Enable bit. Setting this bit enables the partial population of external memories where the entire application bus is routed to a reduced size memory system. The lower half of the SDRAM data bus, bit 0 up to bit UPCTL_M_DW/2-1, is the active portion when Partially Populated memories are enabled.</p> <p>1'b0: Disabled 1'b1: Enabled</p>

**DDR\_PCTL\_MSTAT**

Address: Operational Base + offset (0x0088)

Memory Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2	RO	0x0	<p>self_refresh</p> <p>Indicates if uPCTL, through auto self refresh, has placed the memories in Self Refresh.</p> <p>1'b0: Memory is not in Self Refresh 1'b1: Memory is in Self Refresh</p>
1	RO	0x0	<p>clock_stop</p> <p>Indicates if uPCTL has placed the memories in Clock Stop.</p> <p>1'b0: Memory is not in Clock Stop 1'b1: Memory is in Clock Stop</p>
0	RO	0x0	<p>power_down</p> <p>Indicates if uPCTL has placed the memories in Power Down.</p> <p>1'b0: Memory is not in Power Down 1'b1: Memory is in Power-Down</p>

**DDR\_PCTL\_LPDDR2ZQCFG**

Address: Operational Base + offset (0x008c)

LPDDR2 ZQ Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0xab	<p>zqcl_op</p> <p>Value to drive on memory address bits [19:12] for an automatic hardware generated ZQCL command (LPDDR2). Corresponds to OP7 .. OP0 of Mode Register Write (MRW) command which is used to send ZQCL command to memory.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23:16	RW	0x0a	zqcl_ma Value to drive on memory address bits [11:4] for an automatic hardware generated ZQCL command (LPDDR2). Corresponds to MA7 .. MA0 of Mode Register Write (MRW) command which is used to send ZQCL command to memory.
15:8	RW	0x56	zqcs_op Value to drive on memory address bits [19:12] for an automatic hardware generated ZQCS command (LPDDR2). Corresponds to OP7 .. OPO of Mode Register Write (MRW) command which is used to send ZQCS command to memory.
7:0	RW	0x0a	zqcs_ma Value to drive on memory address bits [11:4] for an automatic hardware generated ZQCS command (LPDDR2). Corresponds to MA7 .. MA0 of Mode Register Write (MRW) command which is used to send ZQCS command to memory.

**DDR\_PCTL\_MCFG1**

Address: Operational Base + offset (0x0090)

Memory Configuration 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	hw_exit_idle_en When this bit is programmed to 1'b1 the c_active_in pin can be used to exit from the automatic clock stop, power down or self-refresh modes.
30:24	RO	0x0	reserved
23:16	RW	0x00	hw_idle Hardware idle period. The c_active output is driven high if the NIF is idle in Access state for hw_idle * 32 * n_clk cycles. The hardware idle function is disabled when hw_idle=0.
15:8	RO	0x0	reserved
7:0	RW	0x00	sr_idle Self Refresh idle period. Memories are placed into Self-Refresh mode if the NIF is idle in Access state for sr_idle * 32 * n_clk cycles. The automatic self refresh function is disabled when sr_idle=0.

**DDR\_PCTL\_DTUPDES**

Address: Operational Base + offset (0x0094)

DTU Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13	RO	0x0	dtu_rd_missing Indicates if one or more read beats of data did not return from memory
12:9	RO	0x0	dtu_eaffl Indicates the number of entries in the FIFO that is holding the log of error addresses for data comparison
8	RO	0x0	dtu_random_error Indicates that the random data generated had some failures when written and read to the memories
7	RO	0x0	dtu_err_b7 Detected at least 1 bit error for bit 7 in the programmable data buffers
6	RO	0x0	dtu_err_b6 Detected at least 1 bit error for bit 6 in the programmable data buffers
5	RO	0x0	dtu_err_b5 Detected at least 1 bit error for bit 5 in the programmable data buffers
4	RO	0x0	dtu_err_b4 Detected at least 1 bit error for bit 4 in the programmable data buffers
3	RO	0x0	dtu_err_b3 Detected at least 1 bit error for bit 3 in the programmable data buffers
2	RO	0x0	dtu_err_b2 Detected at least 1 bit error for bit 2 in the programmable data buffers
1	RO	0x0	dtu_err_b1 Detected at least 1 bit error for bit 1 in the programmable data buffers
0	RO	0x0	dtu_err_b0 Detected at least 1 bit error for bit 0 in the programmable data buffers

**DDR\_PCTL\_DTUNA**

Address: Operational Base + offset (0x0098)

DTU Number of Addresses Created Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	dtu_num_address Indicates the number of addresses that were created on the NIF interface during random data generation

**DDR\_PCTL\_DTUNE**

Address: Operational Base + offset (0x009c)

DTU Number of Errors Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	dtu_num_errors Indicates the number of errors that were detected on the readback of the NIF data during random data generation.

**DDR\_PCTL\_DTUPRD0**

Address: Operational Base + offset (0x00a0)

DTU Parallel Read 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	dtu_allbits_1 Allows all the bit ones from each of the 16 received read bytes to be read in parallel. Used as part of read data eye training where a transition is required to be monitored to train the eye.
15:0	RO	0x0000	dtu_allbits_0 Allows all the bit zeros from each of the 16 received read bytes to be read in parallel. Used as part of read data eye training where a transition is required to be monitored to train the eye.

**DDR\_PCTL\_DTUPRD1**

Address: Operational Base + offset (0x00a4)

DTU Parallel Read 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	dtu_allbits_3 Allows all the bit threes from each of the 16 received read bytes to be read in parallel. Used as part of read data eye training where a transition is required to be monitored to train the eye.
15:0	RO	0x0000	dtu_allbits_2 Allows all the bit twos from each of the 16 received read bytes to be read in parallel. Used as part of read data eye training where a transition is required to be monitored to train the eye.

**DDR\_PCTL\_DTUPRD2**

Address: Operational Base + offset (0x00a8)

DTU Parallel Read 2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	dtu_allbits_5 Allows all the bit fives from each of the 16 received read bytes to be read in parallel. Used as part of read data eye training where a transition is required to be monitored to train the eye.
15:0	RO	0x0000	dtu_allbits_4 Allows all the bit fours from each of the 16 received read bytes to be read in parallel. Used as part of read data eye training where a transition is required to be monitored to train the eye.

**DDR\_PCTL\_DTUPRD3**

Address: Operational Base + offset (0x00ac)

DTU Parallel Read 3 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	dtu_allbits_7 Allows all the bit sevens from each of the 16 received read bytes to be read in parallel. Used as part of read data eye training where a transition is required to be monitored to train the eye.
15:0	RO	0x0000	dtu_allbits_6 Allows all the bit sixes from each of the 16 received read bytes to be read in parallel. Used as part of read data eye training where a transition is required to be monitored to train the eye.

**DDR\_PCTL\_DTUAWDT**

Address: Operational Base + offset (0x00b0)

DTU Address Width Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved
10:9	RW	0x1	number_ranks Number of supported memory ranks. 2'b00: 1 rank 2'b01: 2 ranks 2'b10: 3 ranks 2'b11: 4 ranks
8	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:6	RW	0x2	row_addr_width Width of the memory row address bits. 2'b00: 13 bits wide 2'b01: 14 bits wide 2'b10: 15 bits wide 2'b11: 16 bits wide
5	RO	0x0	reserved
4:3	RW	0x2	bank_addr_width Width of the memory bank address bits. 2'b00: 2 bits wide (4 banks) 2'b01: 3 bits wide (8 banks) Others: Reserved
2	RO	0x0	reserved
1:0	RW	0x0	column_addr_width Width of the memory column address bits. 2'b00: 7 bits wide 2'b01: 8 bits wide 2'b10: 9 bits wide 2'b11: 10 bits wide

**DDR\_PCTL\_TOGCNT1U**

Address: Operational Base + offset (0x00c0)

Toggle Counter 1us Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9:0	RW	0x064	toggle_counter_1u The number of internal timers clock cycles

**DDR\_PCTL\_TINIT**

Address: Operational Base + offset (0x00c4)

t\_init Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8:0	RW	0x0c8	t_init Defines the time period (in us) to hold dfi_cke and dfi_reset_n stable during the memory power up sequence. The value programmed must correspond to at least 200us. The actual time period defined is TINIT * TOGCNT1U * internal timers clock period.

**DDR\_PCTL\_TRSTH**

Address: Operational Base + offset (0x00c8)

## t\_rsth Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9:0	RW	0x000	t_rsth Defines the time period (in us) to hold the dfi_reset_n signal high after it is de-asserted during the DDR3 Power Up/Reset sequence. The value programmed for DDR3 must correspond to minimum 500us of delay. For mDDR and DDR2, this register should be programmed to 0. The actual time period defined is TRSTH * TOGCNT1U * internal timers clock period.

**DDR\_PCTL\_TOGCNT100N**

Address: Operational Base + offset (0x00cc)

Toggle Counter 100ns

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6:0	RW	0x01	toggle_counter_100n The number of internal timers clock cycles

**DDR\_PCTL\_TREFI**

Address: Operational Base + offset (0x00d0)

t\_refi Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x01	t_refi Defines the time period (in 100ns units) of the Refresh interval. The actual time period defined is TREFI * TOGCNT100N * internal timers clock period.

**DDR\_PCTL\_TMRD**

Address: Operational Base + offset (0x00d4)

t\_mrd Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:0	RW	0x1	t_mrd Mode Register Set command cycle time in memory clock cycles. mDDR: Time from MRS to any valid command LPDDR2: Time from MRS (MRW) to any valid command DDR2: Time from MRS to any valid command DDR3: Time from MRS to MRS command mDDR Legal Values: 2 LPDDR2 Legal Values: 5 DDR2 Legal Values: 2..3 DDR3 Legal Values: 2..4

**DDR\_PCTL\_TRFC**

Address: Operational Base + offset (0x00d8)

t\_rfc Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8:0	RW	0x001	t_rfc Refresh to Active/Refresh command time in memory clock cycles. mDDR Legal Values: 7..28 LPDDR2 Legal Values: 15..112 DDR2 Legal Values: 15..131 DDR3 Legal Values: 36.. 374

**DDR\_PCTL\_TRP**

Address: Operational Base + offset (0x00dc)

t\_trp Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:18	RO	0x0	reserved
17:16	RW	0x1	prea_extra Additional cycles required for a Precharge All (PREA) command - in addition to t_rp. In terms of memory clock cycles. mDDR Value: 0 LPDDR2 Value: Value that corresponds (tRPab -tRPpb). Rounded up in terms of memory clock cycles. Values can be 0, 1, 2. DDR2 Value: 1 if 8 Banks, 0 otherwise DDR3 Value: 0
15:4	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x6	t_rp Precharge period in memory clock cycles. For LPDDR2, this should be set to TRPpb. mDDR Legal Values: 2..3 LPDDR2 Legal Values: 3..13 DDR2 Legal Values: 3..7 DDR3 Legal Values: 5..14

**DDR\_PCTL\_TRTW**

Address: Operational Base + offset (0x00e0)

t\_rtw Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x2	t_rtw Read to Write turnaround time in memory clock cycles. mDDR Legal Values: 3..11 LPDDR2 Legal Values: 1..11 DDR2 Legal Values: 2..10 DDR3 Legal Values: 2..10

**DDR\_PCTL\_TAL**

Address: Operational Base + offset (0x00e4)

AL Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x0	t_al Additive Latency in memory clock cycles. For DDR2 this must match the value programmed into the AL field of MR1. For DDR3 this must be 0, CL-1, CL-2 depending whether the AL value in MR1 is 0,1, or 2 respectively. CL is the CAS latency programmed into MR0. For mDDR and LPDDR2, there is no AL field in the mode registers, and this setting should be set to 0 mDDR Legal Values: 0 LPDDR2 Legal Values: 0 DDR2 Legal Values: AL DDR3 Legal Values: 0, CL-1, CL-2 (depending on AL=0,1,2 in MR1)

**DDR\_PCTL\_TCL**

Address: Operational Base + offset (0x00e8)

CL Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x4	t_cl CAS Latency in memory clock cycles. If mDDR/DDR2/DDR3, the uPCTL setting must match the value programmed into the CL field of MR0. If LPDDR2, the uPCTL setting must match RL (ReadLatency), where RL is the value programmed into the "RL & W" field of MR2 mDDR/DDR2/3 Legal Value: CL LPDDR2 Legal Value: RL

**DDR\_PCTL\_TCWL**

Address: Operational Base + offset (0x00ec)

CWL Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x3	t_cwl CAS Write Latency in memory clock cycles. For mDDR, the setting must be 1. For LPDDR2 the setting must match WL (Write Latency), where WL is the value programmed into the "RL & WL" field of MR2. For DDR2 the setting must match CL-1, where CL is the value programmed into the CL field of MR0. For DDR3, the setting must match the value programmed in the memory CWL field of MR2. mDDR Legal Value: 1 LPDDR2 Legal Values: WL DDR2 Legal Value: CL-1 DDR3 Legal Value: CWL

**DDR\_PCTL\_TRAS**

Address: Operational Base + offset (0x00f0)

t\_ras Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:0	RW	0x10	t_ras Activate to Precharge command time in memory clock cycles. mDDR Legal Values: 4..8 LPDDR2 Legal Values: 7..23 DDR2 Legal Values: 8..24 DDR3 Legal Values: 15..38

**DDR\_PCTL\_TRC**

Address: Operational Base + offset (0x00f4)

t\_rc Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RW	0x16	t_rc Row Cycle time in memory clock cycles. Specifies the minimum Activate to Activate distance for accesses to same bank. mDDR Legal Values: 5..11 LPDDR2 Legal Values: 10..36 DDR2 Legal Values: 11..31 DDR3 Legal Values: 20..52

**DDR\_PCTL\_TRCD**

Address: Operational Base + offset (0x00f8)

t\_rcd Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x6	t_rcd Row to Column delay in memory clock cycles. Specifies the minimum Activate to Column distance. mDDR Legal Values: 2..3 LPDDR2 Legal Values: 3..13 DDR2 Legal Values: 3..7 DDR3 Legal Values: 5..14

**DDR\_PCTL\_TRRD**

Address: Operational Base + offset (0x00fc)

t\_rrd Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x4	t_rrd Row-to-Row delay in memory clock cycles. Specifies the minimum Activate-to-Activate distance for consecutive accesses to different banks in the same rank. mDDR Legal Values: 1..2 LPDDR2 Legal Values: 2..6 DDR2 Legal Values: 2..6 DDR3 Legal Values: 4..8

**DDR\_PCTL\_TRTP**

Address: Operational Base + offset (0x0100)

t\_rtp Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x3	t_rtp Read to Precharge time in memory clock cycles. Specifies the minimum distance Read to Precharge for consecutive accesses to same bank. mDDR Value: 0 LPDDR2 Legal Values: 2..4 DDR2 Legal Values: 2..4 DDR3 Legal Values: 3..8

**DDR\_PCTL\_TWR**

Address: Operational Base + offset (0x0104)

t\_wr Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RW	0x06	t_wr Write recovery time in memory clock cycles. When using close page the uPCTL setting must be consistent with the WR field setting of MR0. mDDR Legal Values: 2..3 LPDDR2 Legal Values: 3..8 DDR2 Legal Values: 3..8 DDR3 Legal Values: 6..16

**DDR\_PCTL\_TWTR**

Address: Operational Base + offset (0x0108)

t\_wtr Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x4	t_wtr Write to Read turnaround time, in memory clock cycles. mDDR Legal Values: 1..2 LPDDR2 Legal Values: 2..4 DDR2 Legal Values: 2..4 DDR3 Legal Values: 3..8

**DDR\_PCTL\_TEXSR**

Address: Operational Base + offset (0x010c)

t\_exsr Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9:0	RW	0x001	t_exsr Exit Self Refresh to first valid command delay, in memory clock cycles. For mDDR, this should be programmed to match tXSR. For LPDDR2, this should be programmed to match tXSR. For DDR2, this should be programmed to match tXSRD (SRE to read-related command) as defined by the memory device specification. For DDR3, this should be programmed to match tXSDLL (SRE to a command requiring DLL locked) as defined by the memory device specification. mDDR Legal Values: 17..40 LPDDR2 Legal Values: 17..117 DDR2 Typical Value: 200 DDR3 Typical Value: 512

**DDR\_PCTL\_TXP**

Address: Operational Base + offset (0x0110)

t\_xp Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2:0	RW	0x1	t_xp Exit Power Down to first valid command delay when DLL is on (fast exit), measured in memory clock cycles. Legal Values: 1..7

**DDR\_PCTL\_TXPDLL**

Address: Operational Base + offset (0x0114)

t\_xpdll Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RW	0x00	t_xpdll Exit Power Down to first valid command delay when DLL is off (slow exit), measured in memory clock cycles. mDDR/LPDDR2 Value: 0 DDR2/DDR3 Legal Values: 3..63

**DDR\_PCTL\_TZQCS**

Address: Operational Base + offset (0x0118)

t\_zqcs Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6:0	RW	0x00	t_zqcs SDRAM ZQ Calibration Short period, in memory clock cycles. Should be programmed to match the tZQCS timing value as defined in the memory specification. mDDR Value: 0 LPDDR2 Legal Values: 15..48 DDR2 Value: 0 DDR3 Typical Value: 64

**DDR\_PCTL\_TZQCSI**

Address: Operational Base + offset (0x011c)

t\_zqcsi Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	t_zqcsi SDRAM ZQCS interval, measured in Refresh interval units. The total time period defined is TZQCSI*TREFI * TOGCNT100N * internal timers clock period. Programming a value of 0 in t_zqcsi disables the auto-ZQCS functionality in uPCTL. mDDR Value: 0 LPDDR2 Legal Values: 0..4294967295 DDR2 Value: 0 DDR3 Legal Values: 0..4294967295

**DDR\_PCTL\_TDQS**

Address: Operational Base + offset (0x0120)

t\_dqs Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2:0	RW	0x1	t_dqs Additional data turnaround time in memory clock cycles for accesses to different ranks. Used to increase the distance between column commands to different ranks, allowing more tolerance as the driver source changes on the bidirectional DQS and/or DQ signals. mDDR Legal Values: 1..7 LPDDR2 Legal Values: 1..7 DDR2 Legal Values: 1..7 DDR3 Legal Values: 1..7

**DDR\_PCTL\_TCKSRE**

Address: Operational Base + offset (0x0124)

t\_cksrc Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RW	0x00	t_cksrc In DDR3, this is the time after Self Refresh Entry that CKE is held high before going low. In memory clock cycles. Specifies the clock disable delay after SRE. This should be programmed to match the greatest value between 10ns and 5 memory clock periods. mDDR Value: 0 LPDDR2 Value: 0 DDR2 Value: 0 DDR3 Legal Values: 5..15

**DDR\_PCTL\_TCKSRX**

Address: Operational Base + offset (0x0128)

t\_cksrc Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:0	RW	0x00	t_cksr <p>In DDR3, this is the time (before Self Refresh Exit) that CKE is maintained high before issuing SRX. In memory clock cycles. Specifies the clock stable time before SRX. This should be programmed to match the greatest value between 10ns and 5 memory clock periods.</p> <p>mDDR Value: 0 LPDDR2 Value: 0 DDR2 Value: 0 DDR3 Legal Values: 5..15</p>

**DDR\_PCTL\_TCKE**

Address: Operational Base + offset (0x012c)

t\_cke Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2:0	RW	0x3	t_cke <p>CKE minimum pulse width in memory clock cycles.</p> <p>mDDR Legal Value: 2 LPDDR2 Legal Values: 3 DDR2 Legal Value: 3 DDR3 Legal Values: 3..6</p>

**DDR\_PCTL\_TMOD**

Address: Operational Base + offset (0x0130)

t\_mod Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RW	0x00	t_mod <p>In DDR3 mode, this is the time from MRS to any valid non-MRS command (except DESELECT or NOP) in memory clock cycles.</p> <p>mDDR Value: 0 LPDDR2 Value: 0 DDR2 Value: 0 DDR3 Legal Values: 0..31</p>

**DDR\_PCTL\_TRSTL**

Address: Operational Base + offset (0x0134)

Reset Low Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6:0	RW	0x00	t_rstl Memory Reset Low time, in memory clock cycles. Defines the time period to hold dfi_reset_n signal low during a software driven DDR3 Reset Operation. The value programmed must correspond to at least 100ns of delay. mDDR Value: 0 LPDDR2 Value: 0 DDR2 Value: 0 DDR3 Legal Values: 1..127

**DDR\_PCTL\_TZQCL**

Address: Operational Base + offset (0x0138)

t\_zqcl Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9:0	RW	0x000	t_zqcl SDRAM ZQ Calibration Long period in memory clock cycles. If LPDDR2, should be programmed to tZQCL. If DDR3, should be programmed to match the memory tZQinit timing value for the first ZQCL command during memory initialization; should be programmed to match tZQoper timing value after reset and initialization. mDDR Value: 0 LPDDR2 Legal Values: 60..192 DDR2 Value: 0 DDR3 Legal Values: 0..1023

**DDR\_PCTL\_TMRR**

Address: Operational Base + offset (0x013c)

t\_mrr Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x02	t_mrr Time for a Mode Register Read (MRR command from MCMD)

**DDR\_PCTL\_TCLESR**

Address: Operational Base + offset (0x0140)

## t\_ckesr Timing Register

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3:0	RW	0x4	t_ckesr Minimum CKE low width for Self Refresh entry to exit timing in memory clock cycles. Recommended settings: mDDR: t_ckesr = 0 LPDDR2: t_ckesr = tCKESR setting from memories, rounded up in terms of memory cycles. DDR2: t_ckesr = 0 DDR3: t_ckesr = t_cke + 1 mDDR Value: 0 LPDDR2 Legal Values: 3..8 DDR2 Value: 0 DDR3 Legal Values: 4..7

## DDR\_PCTL\_TDPD

Address: Operational Base + offset (0x0144)

## t\_dpd Timing Register

Bit	Attr	Reset Value	Description
31:10	RO	0x0	reserved
9:0	RW	0x000	t_dpd Minimum Deep Power Down time. Is in terms of us. When a MCMD.DPDE command occurs, TDPD time is waited before MCMD.start_cmd can be cleared. MCMD_cmd_add_del (if any) does not start until TDPD has completed. This ensures TDPD requirement for the memory is not violated. The actual time period defined is TDPD* TOGCNT1U * internal timers clock period. Only applies for mDDR and LPDDR2 as Deep Power Down (DPD) is only valid for these memory types. For mDDR, tDPD=0, while for LPDDR2, tDPD=500 us. For LPDDR2, if 500 us is waited externally by system, then set tDPD=0. mDDR Value: 0 LPDDR2 Legal Values: 0 or 500 DDR2 Legal Value: 0 DDR3 Legal Values: 0

## DDR\_PCTL\_DTUWACTL

Address: Operational Base + offset (0x0200)

## DTU Write Address Control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RW	0x0	dtu_wr_rank Write rank to where data is to be targeted
29	RO	0x0	reserved
28:13	RW	0x0000	dtu_wr_row Write row to where data is to be targeted
12:10	RW	0x0	dtu_wr_bank Write bank to where data is to be targeted
9:0	RW	0x000	dtu_wr_col FWrite column to where data is to be targeted

**DDR\_PCTL\_DTURACTL**

Address: Operational Base + offset (0x0204)

DTU Read Address Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RW	0x0	dtu_rd_rank Read rank from where data comes
29	RO	0x0	reserved
28:13	RW	0x0000	dtu_rd_row Read row from where data comes
12:10	RW	0x0	dtu_rd_bank Read bank from where data comes
9:0	RW	0x000	dtu_rd_col Read column from where data comes

**DDR\_PCTL\_DTUCFG**

Address: Operational Base + offset (0x0208)

DTU Configuration Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:23	RO	0x0	reserved
22:16	RW	0x00	dtu_row_increments Number of times to increment the row address when generating random data, up to a maximum of 127 times.
15	RW	0x0	dtu_wr_multi_rd When set puts the DTU into write once multiple reads mode
14	RW	0x0	dtu_data_mask_en Controls whether random generated data masks are transmitted. Unless enabled all data bytes are written to memory and expected to be read from memory.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13:10	RW	0x0	dtu_target_lane Selects one of the byte lanes for data comparison into the programmable read data buffer
9	RW	0x0	dtu_generate_random Generate transfers using random data, otherwise generate transfers from the programmable write data buffers.
8	RW	0x0	dtu_incr_banks When the column address rolls over increment the bank address until we reach and conclude bank 7.
7	RW	0x0	dtu_incr_cols Increment the column address until we saturate. Return to zero if DTUCFG.dtu_incr_banks is set to 1 and we are not at bank 7.
6:1	RW	0x00	dtu_nalen Length of the NIF transfer sequence that is passed through the uPCTL for each created address.
0	RW	0x0	dtu_enable When set, allows the DTU module to take ownership of the NIF interface: 1'b1: DTU enabled 1'b0: DTU disabled

**DDR\_PCTL\_DTUECTL**

Address: Operational Base + offset (0x020c)

DTU Execute Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2	RWSC	0x0	wr_multi_rd_RST When set, resets the DTU in write once multiple reads mode, to allow a new write to be performed. This bit automatically clears.
1	RWSC	0x0	run_error_reports When set, initiates the calculation of the error status bits. This bit automatically clears when the re-calculation is done. This is only used in debug mode to verify the comparison logic.
0	RWSC	0x0	run_dtu When set, initiates the running of the DTU read and write transfer. This bit automatically clears when the transfers are completed.

**DDR\_PCTL\_DTUWDO**

Address: Operational Base + offset (0x0210)

DTU Write Data #0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	dtu_wr_byte3 Write data byte
23:16	RW	0x00	dtu_wr_byte2 Write data byte
15:8	RW	0x00	dtu_wr_byte1 Write data byte
7:0	RW	0x00	dtu_wr_byte0 Write data byte

### **DDR\_PCTL\_DTUWD1**

Address: Operational Base + offset (0x0214)

DTU Write Data #1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	dtu_wr_byte7 Write data byte
23:16	RW	0x00	dtu_wr_byte6 Write data byte
15:8	RW	0x00	dtu_wr_byte5 Write data byte
7:0	RW	0x00	dtu_wr_byte4 Write data byte

### **DDR\_PCTL\_DTUWD2**

Address: Operational Base + offset (0x0218)

DTU Write Data #2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	dtu_wr_byte11 Write data byte
23:16	RW	0x00	dtu_wr_byte10 Write data byte
15:8	RW	0x00	dtu_wr_byte9 Write data byte
7:0	RW	0x00	dtu_wr_byte8 Write data byte

### **DDR\_PCTL\_DTUWD3**

Address: Operational Base + offset (0x021c)

DTU Write Data #3 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	dtu_wr_byte15 Write data byte
23:16	RW	0x00	dtu_wr_byte14 Write data byte
15:8	RW	0x00	dtu_wr_byte13 Write data byte
7:0	RW	0x00	dtu_wr_byte12 Write data byte

**DDR\_PCTL\_DTUWDM**

Address: Operational Base + offset (0x0220)

DTU Write Data Mask Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	dm_wr_byte0 Write data mask bit, one bit for each byte. Each bit should be 0 for a byte lane that contains valid write data.

**DDR\_PCTL\_DTURDO**

Address: Operational Base + offset (0x0224)

DTU Read Data #0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x00	dtu_rd_byte3 Read byte
23:16	RO	0x00	dtu_rd_byte2 Read byte
15:8	RO	0x00	dtu_rd_byte1 Read byte
7:0	RO	0x00	dtu_rd_byte0 Read byte

**DDR\_PCTL\_DTURD1**

Address: Operational Base + offset (0x0228)

DTU Read Data #1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x00	dtu_rd_byte7 Read byte
23:16	RO	0x00	dtu_rd_byte6 Read byte

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:8	RO	0x00	dtu_rd_byte5 Read byte
7:0	RO	0x00	dtu_rd_byte4 Read byte

**DDR\_PCTL\_DTURD2**

Address: Operational Base + offset (0x022c)

DTU Read Data #2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x00	dtu_rd_byte11 Read byte
23:16	RO	0x00	dtu_rd_byte10 Read byte
15:8	RO	0x00	dtu_rd_byte9 Read byte
7:0	RO	0x00	dtu_rd_byte8 Read byte

**DDR\_PCTL\_DTURD3**

Address: Operational Base + offset (0x0230)

DTU Read Data #3 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x00	dtu_rd_byte15 Read byte
23:16	RO	0x00	dtu_rd_byte14 Read byte
15:8	RO	0x00	dtu_rd_byte13 Read byte
7:0	RO	0x00	dtu_rd_byte12 Read byte

**DDR\_PCTL\_DTULFSRWD**

Address: Operational Base + offset (0x0234)

DTU LFSR Seed for Write Data Generation Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dtu_lfsr_wseed This is the initial seed for the random write data generation LFSR (linear feedback shift register), shared with the write mask generation.

**DDR\_PCTL\_DTULFSRRD**

Address: Operational Base + offset (0x0238)

DTU LFSR Seed for Read Data Generation Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dtu_lfsr_rseed This is the initial seed for the random read data generation LFSR (linear feedback shift register), this is shared with the read mask generation. The read data mask is reconstructed the same as the write data mask was created, allowing the "on the fly comparison" ignore bytes which were not written.

**DDR\_PCTL\_DTUAEAF**

Address: Operational Base + offset (0x023c)

DTU Error Address FIFO Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	ea_rank Indicates the rank that the error occurred in during random data generation. There could be a number of entries in this FIFO. If FIFO is empty one reads zeroes.
29	RO	0x0	reserved
28:13	RO	0x0000	ea_row Indicates the row that the error occurred in during random data generation. There could be a number of entries in this FIFO. If FIFO is empty one reads zeroes.
12:10	RO	0x0	ea_bank Indicates the bank that the error occurred in during random data generation. There could be a number of entries in this FIFO. If FIFO is empty one reads zeroes.
9:0	RO	0x000	ea_column Indicates the column address that the error occurred in during random data generation. There could be a number of entries in this FIFO. If FIFO is empty one reads zeroes.

**DDR\_PCTL\_DFITCTRLDELAY**

Address: Operational Base + offset (0x0240)

DFI tctrl\_delay Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x2	tctrl_delay Specifies the number of DFI clock cycles after an assertion or deassertion of the DFI control signals that the control signals at the PHY-DRAM interface reflect the assertion or de-assertion. If the DFI clock and the memory clock are not phase-aligned, this timing parameter should be rounded up to the next integer value.

**DDR\_PCTL\_DFIODTCFG**

Address: Operational Base + offset (0x0244)

DFI ODT Configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28	RW	0x0	rank3_odt_default Default ODT value of rank 3 when there is no read/write activity
27	RW	0x0	rank3_odt_write_sel Enable/disable ODT for rank 3 when a write access is occurring on this rank
26	RW	0x0	rank3_odt_write_nse Enable/disable ODT for rank 3 when a write access is occurring on a different rank
25	RW	0x0	rank3_odt_read_sel Enable/disable ODT for rank 3 when a read access is occurring on this rank
24	RW	0x0	rank3_odt_read_nsel Enable/disable ODT for rank 3 when a read access is occurring on a different rank
23:21	RO	0x0	reserved
20	RW	0x0	rank2_odt_default Default ODT value of rank 2 when there is no read/write activity
19	RW	0x0	rank2_odt_write_sel Enable/disable ODT for rank 2 when a write access is occurring on this rank
18	RW	0x0	rank2_odt_write_nse Enable/disable ODT for rank 2 when a write access is occurring on a different rank
17	RW	0x0	rank2_odt_read_sel Enable/disable ODT for rank 2 when a read access is occurring on this rank

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16	RW	0x0	rank2_odt_read_nsel Enable/disable ODT for rank 2 when a read access is occurring on a different rank
15:13	RO	0x0	reserved
12	RW	0x0	rank1_odt_default Default ODT value of rank 1 when there is no read/write activity
11	RW	0x0	rank1_odt_write_sel Enable/disable ODT for rank 1 when a write access is occurring on this rank
10	RW	0x0	rank1_odt_write_nse Enable/disable ODT for rank 1 when a write access is occurring on a different rank
9	RW	0x0	rank1_odt_read_sel Enable/disable ODT for rank 1 when a read access is occurring on this rank
8	RW	0x0	rank1_odt_read_nsel Enable/disable ODT for rank 1 when a read access is occurring on a different rank
7:5	RO	0x0	reserved
4	RW	0x0	rank0_odt_default Default ODT value of rank 0 when there is no read/write activity
3	RW	0x0	rank0_odt_write_sel Enable/disable ODT for rank 0 when a write access is occurring on this rank
2	RW	0x0	rank0_odt_write_nse Enable/disable ODT for rank 0 when a write access is occurring on a different rank
1	RW	0x0	rank0_odt_read_sel Enable/disable ODT for rank 0 when a read access is occurring on this rank
0	RW	0x0	rank0_odt_read_nsel Enable/disable ODT for rank 0 when a read access is occurring on a different rank

**DDR\_PCTL\_DFIODTCFG1**

Address: Operational Base + offset (0x0248)

DFI ODT Timing Configuration 1 (for Latency and Length)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:27	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
26:24	RW	0x6	odt_len.bl8_r ODT length for BL8 read transfers Length of dfi_odt signal for BL8 reads. This is in terms of SDR cycles. For BL4 reads, the length of dfi_odt is always 2 cycles shorter than the value in this register field.
23:19	RO	0x0	reserved
18:16	RW	0x6	odt_len.bl8_w ODT length for BL8 write transfers Length of dfi_odt signal for BL8 writes. This is in terms of SDR cycles. For BL4 writes, the length of dfi_odt is always 2 cycles shorter than the value in this register field.
15:13	RO	0x0	reserved
12:8	RW	0x00	odt_lat.r ODT latency for reads Latency after a read command that dfi_odt is set. This is in terms of SDR cycles.
7:5	RO	0x0	reserved
4:0	RW	0x00	odt_lat.w ODT latency for writes Latency after a write command that dfi_odt is set. This is in terms of SDR cycles.

**DDR\_PCTL\_DFIODTRANKMAP**

Address: Operational Base + offset (0x024c)

DFI ODT Rank Mapping

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:12	RW	0x8	odt_rank_map3 Rank mapping for dfi_odt[3] Determines whether dfi_odt[3] should be asserted Bit 15 = 1: dfi_odt[3] will be asserted to terminate rank 3 Bit 14 = 1: dfi_odt[3] will be asserted to terminate rank 2 Bit 13 = 1: dfi_odt[3] will be asserted to terminate rank 1 Bit 12 = 1: dfi_odt[3] will be asserted to terminate rank 0 This field exists only if UPCTL_M_NRANKS = 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:8	RW	0x4	<p>odt_rank_map2</p> <p>Rank mapping for dfi_odt[2]</p> <p>Determines which rank access(es) will cause dfi_odt[2] to be asserted</p> <p>Bit 11 = 1: dfi_odt[2] will be asserted to terminate rank 3</p> <p>Bit 10 = 1: dfi_odt[2] will be asserted to terminate rank 2</p> <p>Bit 9 = 1: dfi_odt[2] will be asserted to terminate rank 1</p> <p>Bit 8 = 1: dfi_odt[2] will be asserted to terminate rank 0</p> <p>This field exists only if UPCTL_M_NRANKS = 4</p>
7:4	RW	0x2	<p>odt_rank_map1</p> <p>Rank mapping for dfi_odt[1]</p> <p>Determines which rank access(es) will cause dfi_odt[1] to be asserted</p> <p>Bit 7 = 1: dfi_odt[1] will be asserted to terminate rank 3</p> <p>Bit 6 = 1: dfi_odt[1] will be asserted to terminate rank 2</p> <p>Bit 5 = 1: dfi_odt[1] will be asserted to terminate rank 1</p> <p>Bit 4 = 1: dfi_odt[1] will be asserted to terminate rank 0</p>
3:0	RW	0x1	<p>odt_rank_map0</p> <p>Rank mapping for dfi_odt[0]</p> <p>Determines which rank access(es) will cause dfi_odt[0] to be asserted</p> <p>Bit 3 = 1: dfi_odt[0] will be asserted to terminate rank 3</p> <p>Bit 2 = 1: dfi_odt[0] will be asserted to terminate rank 2</p> <p>Bit 1 = 1: dfi_odt[0] will be asserted to terminate rank 1</p> <p>Bit 0 = 1: dfi_odt[0] will be asserted to terminate rank 0</p>

**DDR\_PCTL\_DFITPHYWRDATA**

Address: Operational Base + offset (0x0250)

DFI tphy\_wrdata Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:0	RW	0x01	tphy_wrdata Specifies the number of DFI clock cycles between when the dfi_wrdata_en signal is asserted to when the associated write data is driven on the dfi_wrdata signal. This has no impact on performance, only adjusts the relative time between enable and data transfer.

**DDR\_PCTL\_DFITPHYWRLAT**

Address: Operational Base + offset (0x0254)

DFI tphy\_wrlat Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RW	0x01	tphy_wrlat Specifies the number of DFI clock cycles between when a write command is sent on the DFI control interface and when the dfi_wrdata_en signal is asserted.

**DDR\_PCTL\_DFITRDDATAEN**

Address: Operational Base + offset (0x0260)

DFI trddata\_en Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RW	0x01	trddata_en Specifies the number of DFI clock cycles from the assertion of a read command on the DFI to the assertion of the dfi_rddata_en signal.

**DDR\_PCTL\_DFITPHYRDLAT**

Address: Operational Base + offset (0x0264)

DFI tphy\_rdlat Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RW	0x0f	tphy_rdlat Specifies the maximum number of DFI clock cycles allowed from the assertion of the dfi_rddata_en signal to the assertion of the dfi_rddata_valid signal.

**DDR\_PCTL\_DFITPHYUPDTYPEO**

Address: Operational Base + offset (0x0270)

## DFI tphyupd\_type0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x010	tphyupd_type0 Specifies the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type = 0x0. The dfi_phyupd_req signal may de-assert at any cycle after the assertion of the dfi_phyupd_ack signal.

**DDR\_PCTL\_DFITPHYUPDTYPE1**

Address: Operational Base + offset (0x0274)

## DFI tphyupd\_type1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x010	tphyupd_type1 Specifies the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type = 0x1. The dfi_phyupd_req signal may de-assert at any cycle after the assertion of the dfi_phyupd_ack signal.

**DDR\_PCTL\_DFITPHYUPDTYPE2**

Address: Operational Base + offset (0x0278)

## DFI tphyupd\_type2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x010	tphyupd_type2 Specifies the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type = 0x2. The dfi_phyupd_req signal may de-assert at any cycle after the assertion of the dfi_phyupd_ack signal.

**DDR\_PCTL\_DFITPHYUPDTYPE3**

Address: Operational Base + offset (0x027c)

## DFI tphyupd\_type3 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:0	RW	0x010	tphyupd_type3 Specifies the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type = 0x3. The dfi_phyupd_req signal may de-assert at any cycle after the assertion of the dfi_phyupd_ack signal.

**DDR\_PCTL\_DFITCTRLUPDMIN**

Address: Operational Base + offset (0x0280)

DFI tctrlupd\_min Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0010	tctrlupd_min Specifies the minimum number of DFI clock cycles that the dfi_ctrlupd_req signal must be asserted.

**DDR\_PCTL\_DFITCTRLUPDMAX**

Address: Operational Base + offset (0x0284)

DFI tctrlupd\_max Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0040	tctrlupd_max Specifies the maximum number of DFI clock cycles that the dfi_ctrlupd_req signal can assert.

**DDR\_PCTL\_DFITCTRLUPDDLY**

Address: Operational Base + offset (0x0288)

DFI tctrlupddly Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x8	tctrlupd_dly Delay in DFI clock cycles between time a uPCTL-initiated update could be started and time uPCTL-initiated update actually starts (dfi_ctrlupd_req going high).

**DDR\_PCTL\_DFIUPDCFG**

Address: Operational Base + offset (0x0290)

DFI Update Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x1	dfi_phyupd_en Enables the support for acknowledging PHY-initiated updates: 1'b0 = Disabled 1'b1 = Enabled
0	RW	0x1	dfi_ctrlupd_en Enables the generation of uPCTL-initiated updates 1'b0: Disabled 1'b1: Enabled

**DDR\_PCTL\_DFITREFMSKI**

Address: Operational Base + offset (0x0294)

DFI Masked Refresh Interval

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x00	trefmski Time period of the masked Refresh interval. This value is only used if TREFI==0. Defines the time period (in 100ns units) of the masked Refresh (REFMSK) interval. The actual time period defined is DFITREFMSKI* TOGCNT100N * internal timers clock period.

**DDR\_PCTL\_DFITCTRLUPDI**

Address: Operational Base + offset (0x0298)

DFI tctrlupd\_interval Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	tctrlupd_interval DFI uPCTL-initiated updates interval, measured in terms of Refresh interval units. If TREFI != 0, the time period is defined as DFITCTRLUPDI*TREFI * TOGCNT100N * internal timers clock period. If TREFI == 0 and DFITREFMSKI != 0, the period changes to DFITCTRLUPDI * DFITREFMSKI * TOGCNT100N * internal timers clock period. Programming a value of 0 is the same as programming a value of 1; for instance, a uPCTL-initiated update occurs every Refresh interval.

**DDR\_PCTL\_DFITRCFG0**

Address: Operational Base + offset (0x02ac)

DFI Training Configuration 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19:16	RW	0x0	dfi_wrlvl_rank_sel Determines the value to drive on the output signal dfi_wrlvl_cs_n. The value on dfi_wrlvl_cs_n is the inverse of the setting in this field.
15:13	RO	0x0	reserved
12:4	RW	0x000	dfi_rdlvl_edge Determines the value to drive on the output signal dfi_rdlvl_edge. The value on dfi_rdlvl_edge is the same as the setting in this field.
3:0	RW	0x0	dfi_rdlvl_rank_sel Determines the value to drive on the output signal dfi_rdlvl_cs_n. The value on dfi_rdlvl_cs_n is the inverse of the setting in this field.

**DDR\_PCTL\_DFITRSTAT0**

Address: Operational Base + offset (0x02b0)

DFI Training Status 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:18	RO	0x0	reserved
17:16	RO	0x0	dfi_wrlvl_mode Reports the value of the input signal dfi_wrlvl_mode
15:10	RO	0x0	reserved
9:8	RO	0x0	dfi_rdlvl_gate_mode Reports the value of the input signal dfi_rdlvl_gate_mode
7:2	RO	0x0	reserved
1:0	RO	0x0	dfi_rdlvl_mode Reports the value of the input signal dfi_rdlvl_mode

**DDR\_PCTL\_DFITRWRLVLEN**

Address: Operational Base + offset (0x02b4)

DFI Training dfi\_wrlvl\_en Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8:0	RW	0x000	dfi_wrlvl_en Determines the value to drive on the output signal dfi_wrlvl_en

**DDR\_PCTL\_DFITRRDLVLEN**

Address: Operational Base + offset (0x02b8)

DFI Training dfi\_rdlvl\_en Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8:0	RW	0x000	dfi_rdlvl_en Determines the value to drive on the output signal dfi_rdlvl_en

**DDR\_PCTL\_DFITRRDLVLGATEEN**

Address: Operational Base + offset (0x02bc)

DFI Training dfi\_rdlvl\_gate\_en Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8:0	RW	0x000	dfi_rdlvl_gate_en Determines the value to drive on the output signal dfi_rdlvl_gate_en

**DDR\_PCTL\_DFISTSTAT0**

Address: Operational Base + offset (0x02c0)

DFI Status Status 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:16	RO	0x000	dfi_data_byte_disable Reports the value of the output signal dfi_data_byte_disable
15:6	RO	0x0	reserved
5:4	RO	0x0	dfi_freq_ratio Reports the value of the output signal dfi_freq_ratio
3:2	RO	0x0	reserved
1	RO	0x0	dfi_init_start Reports the value of the output signal dfi_init_start
0	RO	0x0	dfi_init_complete Reports the value of the input signal dfi_init_complete

**DDR\_PCTL\_DFISTCFG0**

Address: Operational Base + offset (0x02c4)

DFI Status Configuration 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2	RW	0x0	<p>dfi_data_byte_disable_en Enables the driving of the dfi_data_byte_disable signal. The value driven on dfi_data_byte_disable is dependent on the setting of PPCFG register.</p> <p>1'b0: Drive dfi_data_byte_disable to default value of all zeroes 1'b1: Drive dfi_data_byte_disable according to value as defined by PPCFG register setting</p> <p><i>Note: should be set to 1'b1 only after PPCFG is correctly set</i></p>
1	RW	0x0	<p>dfi_freq_ratio_en Enables the driving of the dfi_freq_ratio signal. When enabled, the dfi_freq_ratio value driven is dependent on configuration parameter UPCTL_FREQ_RATIO: 2'b00 is driven when UPCTL_FREQ_RATIO=1; 2'b01 is driven when UPCTL_FREQ_RATIO=2.</p> <p>1'b0: Drive dfi_freq_ratio to default value of 2'b00 1'b1: Drive dfi_freq_ratio value according to how configuration parameter is set</p>
0	RW	0x0	<p>dfi_init_start Sets the value of the dfi_init_start signal</p> <p>1'b0: dfi_init_start is driven low 1'b1: dfi_init_start is driven high</p>

**DDR\_PCTL\_DFISTCFG1**

Address: Operational Base + offset (0x02c8)

DFI Status Configuration 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	<p>dfi_dram_clk_disable_en_dpd Enables support of the dfi_dram_clk_disable signal with Deep Power Down (DPD). DPD is only for mDDR/LPDDR2.</p> <p>1'b0: Disable dfi_dram_clk_disable support in relation to DPD 1'b1: Enable dfi_dram_clk_disable support in relation to DPD</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	dfi_dram_clk_disable_en Enables support of the dfi_dram_clk_disable signal with Self Refresh (SR). 1'b0: Disable dfi_dram_clk_disable support in relation to SR 1'b1: Enable dfi_dram_clk_disable support in relation to SR

**DDR\_PCTL\_DFITDRAMCLKEN**

Address: Operational Base + offset (0x02d0)

DFI tdrum\_clk\_enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x2	tdram_clk_enable Specifies the number of DFI clock cycles from the de-assertion of the dfi_dram_clk_disable signal on the DFI until the first valid rising edge of the clock to the DRAM memory devices, at the PHY-DRAM boundary. If the DFI clock and the memory clock are not phase aligned, this timing parameter should be rounded up to the next integer value.

**DDR\_PCTL\_DFITDRAMCLKDIS**

Address: Operational Base + offset (0x02d4)

DFI tdrum\_clk\_disable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x2	tdram_clk_disable Specifies the number of DFI clock cycles from the assertion of the dfi_dram_clk_disable signal on the DFI until the clock to the DRAM memory devices, at the PHY-DRAM boundary, maintains a low value. If the DFI clock and the memory clock are not phase aligned, this timing parameter should be rounded up to the next integer value.

**DDR\_PCTL\_DFISTCFG2**

Address: Operational Base + offset (0x02d8)

DFI Status Configuration 2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	parity_en Enables the DFI parity generation feature (driven on output signal dfi_parity_in) 1'b0: Disable DFI parity generation 1'b1: Enable DFI parity generation
0	RW	0x0	parity_intr_en Enable interrupt generation for DFI parity error (from input signal dfi_parity_error) 1'b0: Disable interrupt 1'b1: Enable interrupt

**DDR\_PCTL\_DFISTPARCLR**

Address: Operational Base + offset (0x02dc)

DFI Status Parity Clear Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RWSC	0x0	parity_log_clr Set this bit to 1'b1 to clear the DFI Status Parity Log register (DFISTPARLOG) 1'b0: Do not clear DFI status Parity Log register 1'b1: Clear DFI status Parity Log register
0	RWSC	0x0	parity_intr_clr Set this bit to 1'b1 to clear the interrupt generated by an DFI parity error (as enabled by DFISTCFG2.parity_intr_en). It also clears the INTRSTAT.parity_intr register field. It is automatically cleared by hardware when the interrupt has been cleared.

**DDR\_PCTL\_DFISTPARLOG**

Address: Operational Base + offset (0x02e0)

DFI Status Parity Log Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	parity_err_cnt Increments any time the DFI parity logic detects a parity error(s) (on dfi_parity_error)

**DDR\_PCTL\_DFLPCFG0**

Address: Operational Base + offset (0x02f0)

DFI Low Power Configuration 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RW	0x0	<p>dfi_lp_wakeup_dpd</p> <p>Value to drive on dfi_lp_wakeup signal when Deep Power Down mode is entered.</p> <p>Determines the DFI's tlp_wakeup time:</p> <p>4'b0000: 16 cycles          4'b0001: 32 cycles          4'b0010: 64 cycles          4'b0011: 128 cycles          4'b0100: 256 cycles          4'b0101: 512 cycles          4'b0110: 1024 cycles          4'b0111: 2048 cycles          4'b1000: 4096 cycles          4'b1001: 8192 cycles          4'b1010: 16384 cycles          4'b1011: 32768 cycles          4'b1100: 65536 cycles          4'b1101: 131072 cycles          4'b1110: 262144 cycles          4'b1111: Unlimited</p>
27:25	RO	0x0	reserved
24	RW	0x0	<p>dfi_lp_en_dpd</p> <p>Enables DFI Low Power interface handshaking during Deep Power Down Entry/Exit</p> <p>1'b0: Disabled          1'b1: Enabled</p>
23:20	RO	0x0	reserved
19:16	RW	0x7	<p>dfi_tlp_resp</p> <p>Setting for tlp_resp time.</p> <p>Same value is used for both Power Down and Self refresh and Deep Power Down modes.</p> <p>DFI 2.1 specification, recommends using value of 7 always.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:12	RW	0x0	<p>dfi_lp_wakeup_sr Value to drive on dfi_lp_wakeup signal when Self Refresh mode is entered. Determines the DFI's tlp_wakeup time:</p> <p>4'b0000: 16 cycles 4'b0001: 32 cycles 4'b0010: 64 cycles 4'b0011: 128 cycles 4'b0100: 256 cycles 4'b0101: 512 cycles 4'b0110: 1024 cycles 4'b0111: 2048 cycles 4'b1000: 4096 cycles 4'b1001: 8192 cycles 4'b1010: 16384 cycles 4'b1011: 32768 cycles 4'b1100: 65536 cycles 4'b1101: 131072 cycles 4'b1110: 262144 cycles 4'b1111: Unlimited</p>
11:9	RO	0x0	reserved
8	RW	0x0	<p>dfi_lp_en_sr Enables DFI Low Power interface handshaking during Self Refresh Entry/Exit 1'b0: Disabled 1'b1: Enabled</p>

Bit	Attr	Reset Value	Description
7:4	RW	0x0	<p>dfi_lp_wakeup_pd Value to drive on dfi_lp_wakeup signal when Power Down mode is entered. Determines the DFI's tlp_wakeup time:</p> <ul style="list-style-type: none"> <li>4'b0000: 16 cycles</li> <li>4'b0001: 32 cycles</li> <li>4'b0010: 64 cycles</li> <li>4'b0011: 128 cycles</li> <li>4'b0100: 256 cycles</li> <li>4'b0101: 512 cycles</li> <li>4'b0110: 1024 cycles</li> <li>4'b0111: 2048 cycles</li> <li>4'b1000: 4096 cycles</li> <li>4'b1001: 8192 cycles</li> <li>4'b1010: 16384 cycles</li> <li>4'b1011: 32768 cycles</li> <li>4'b1100: 65536 cycles</li> <li>4'b1101: 131072 cycles</li> <li>4'b1110: 262144 cycles</li> <li>4'b1111: Unlimited</li> </ul>
3:1	RO	0x0	reserved
0	RW	0x0	<p>dfi_lp_en_pd Enables DFI Low Power interface handshaking during Power Down Entry/Exit</p> <ul style="list-style-type: none"> <li>1'b0: Disabled</li> <li>1'b1: Enabled</li> </ul>

**DDR\_PCTL\_DFITRWRLVLRESP0**

Address: Operational Base + offset (0x0300)

DFI Training dfi\_wrlvl\_resp Status 0 Register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	<p>dfi_wrlvl_resp0 Reports the status of the dfi_wrlvl_resp[31:0] signal</p>

**DDR\_PCTL\_DFITRWRLVLRESP1**

Address: Operational Base + offset (0x0304)

DFI Training dfi\_wrlvl\_resp Status 1 Register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	<p>dfi_wrlvl_resp1 Reports the status of the dfi_wrlvl_resp[63:32] signal</p>

**DDR\_PCTL\_DFITRWRLVLRESP2**

Address: Operational Base + offset (0x0308)

DFI Training dfi\_wrlvl\_resp Status 2 Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0x00	dfi_wrlvl_resp2 Reports the status of the dif_wrlvl_resp[71:64] signal

#### DDR\_PCTL\_DFITRRDLVLRESP0

Address: Operational Base + offset (0x030c)

DFI Training dfi\_rdlvl\_resp Status 0 Register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	dfi_rdlvl_resp0 Reports the status of the dif_rdlvl_resp[31:0] signal

#### DDR\_PCTL\_DFITRRDLVLRESP1

Address: Operational Base + offset (0x0310)

DFI Training dfi\_rdlvl\_resp Status 1 Register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	dfi_rdlvl_resp1 Reports the status of the dif_rdlvl_resp[63:32] signal

#### DDR\_PCTL\_DFITRRDLVLRESP2

Address: Operational Base + offset (0x0314)

DFI Training dfi\_rdlvl\_resp Status 2 Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0x00	dfi_rdlvl_resp2 Reports the status of the dif_rdlvl_resp[71:64] signal

#### DDR\_PCTL\_DFITRWRLVLDelay0

Address: Operational Base + offset (0x0318)

DFI Training dfi\_wrlvl\_delay Configuration 0 Register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	dfi_wrlvl_delay0 Sets the value to be driven on the signal dfi_wrlvl_delay_x[31:0]

#### DDR\_PCTL\_DFITRWRLVLDelay1

Address: Operational Base + offset (0x031c)

DFI Training dfi\_wrlvl\_delay Configuration 1 Register

Bit	Attr	Reset Value	Description

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi_wrlvl_delay1 Sets the value to be driven on the signal dfi_wrlvl_delay_x[63:32]

**DDR\_PCTL\_DFITRWRLVLDELAY2**

Address: Operational Base + offset (0x0320)

DFI Training dfi\_wrlvl\_delay Configuration 2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x00	dfi_wrlvl_delay2 Sets the value to be driven on the signal dfi_wrlvl_delay_x[71:64]

**DDR\_PCTL\_DFITRRDLVLDELAY0**

Address: Operational Base + offset (0x0324)

DFI Training dfi\_rdlvl\_delay Configuration 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi_rdlvl_delay0 Sets the value to be driven on the signal dfi_rdlvl_delay_x[31:0]

**DDR\_PCTL\_DFITRRDLVLDELAY1**

Address: Operational Base + offset (0x0328)

DFI Training dfi\_rdlvl\_delay Configuration 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi_rdlvl_delay1 Sets the value to be driven on the signal dfi_rdlvl_delay_x[63:32]

**DDR\_PCTL\_DFITRRDLVLDELAY2**

Address: Operational Base + offset (0x032c)

DFI Training dfi\_rdlvl\_delay Configuration 2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x00	dfi_rdlvl_delay2 Sets the value to be driven on the signal dfi_rdlvl_delay_x[71:64]

**DDR\_PCTL\_DFITRRDLVLGATEDELAY0**

Address: Operational Base + offset (0x0330)

DFI Training dfi\_rdlvl\_gate\_delay Configuration 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi_rdlvl_gate_delay0 Sets the value to be driven on the signal dfi_rdlvl_gate_delay_x[31:0]

**DDR\_PCTL\_DFITRRDLVLGATEDELAY1**

Address: Operational Base + offset (0x0334)

DFI Training dfi\_rdlvl\_gate\_delay Configuration 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dfi_rdlvl_gate_delay1 Sets the value to be driven on the signal dfi_rdlvl_gate_delay_x[63:32]

**DDR\_PCTL\_DFITRRDLVLGATEDELAY2**

Address: Operational Base + offset (0x0338)

DFI Training dfi\_rdlvl\_gate\_delay Configuration 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x00	dfi_rdlvl_gate_delay2 Sets the value to be driven on the signal dfi_rdlvl_gate_delay_x[71:64]

**DDR\_PCTL\_DFITRCMD**

Address: Operational Base + offset (0x033c)

DFI Training Command Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RWSC	0x0	dfitrcmd_start DFI Training Command Start. When this bit is set to 1, the command operation defined in the dfitrcmd_opcode field is started. This bit is automatically cleared by the uPCTL after the command is finished. The application can poll this bit to determine when uPCTL is ready to accept another command. This bit cannot be cleared to 1'b0 by software.
30:13	RO	0x0	reserved
12:4	RW	0x000	dfitrcmd_en DFI Training Command Enable. Selects which bits of chosen DFI Training command to drive to 1'b1.
3:2	RO	0x0	reserved
1:0	RW	0x0	dfitrcmd_opcode DFI Training Command Opcode. Select which DFI Training command to generate for one n_clk cycle: 2'b00: dfi_wrlvl_load 2'b01: dfi_wrlvl_strobe 2'b10: dfi_rdlvl_load 2'b11: Reserved.

**DDR\_PCTL\_IPVR**

Address: Operational Base + offset (0x03f8)

IP Version Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	ip_version ASCII value for each number in the version, followed by a *.

**DDR\_PCTL\_IPTR**

Address: Operational Base + offset (0x03fc)

IP Type Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x44574300	ip_type Contains the IP's identification code, which is an ASCII value to identify the component and it is currently set to the string "DWC". This value never changes.

**DDR\_PUBL\_RIDR**

Address: Operational Base + offset (0x0000)

Revision Identification Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x00	UDRID User-Defined Revision ID General purpose revision identification set by the user
23:20	RO	0x1	PHYMJR PHY Major Revision Indicates major revision of the PHY such addition of the features that make the new version not compatible with previous versions
19:16	RO	0x0	PHYMDR PHY Moderate Revision Indicates moderate revision of the PHY such as addition of new features. Normally the new version is still compatible with previous versions.
15:12	RO	0x0	PHYMNR PHY Minor Revision Indicates minor update of the PHY such as bug fixes. Normally no new features are included.
11:8	RO	0x1	PUBMJR PUB Major Revision Indicates major revision of the PUB such addition of the features that make the new version not compatible with previous versions.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:4	RO	0x4	PUBMDR PUB Moderate Revision Indicates moderate revision of the PUB such as addition of new features. Normally the new version is still compatible with previous versions.
3:0	RO	0x0	PUBMNR PUB Minor Revision Indicates minor update of the PUB such as bug fixes. Normally no new features are included.

**DDR\_PUBL\_PIR**

Address: Operational Base + offset (0x0004)

PHY Initialization Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RWSC	0x0	INITBYP Initialization Bypass Bypasses or stops, if set, all initialization routines currently running, including PHY initialization, DRAM initialization, and PHY training. Initialization may be triggered manually using INIT and the other relevant bits of the PIR register. This bit is self-clearing.
30	RWSC	0x0	ZCALBYP Impedance Calibration Bypass Bypasses or stops, if set, impedance calibration of all ZQ control blocks that automatically triggers after reset. Impedance calibration may be triggered manually using INIT and ZCAL bits of the PIR register. This bit is self-clearing.
29	RWSC	0x0	LOCKBYP DLL Lock Bypass Bypasses or stops, if set, the waiting of DLLs to lock. DLL lock wait is automatically triggered after reset. DLL lock may be triggered manually using INIT and DLLLOCK bits of the PIR register. This bit is self-clearing.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
28	R/WSC	0x0	<p>CLRSR Clear Status Registers</p> <p>A write of '1' to this bit will clear (reset to '0' all status registers, including PGSR and DXnGSR. The clear status register bit is self-clearing.</p> <p>This bit is primarily for debug purposes and is typically not needed during normal functional operation. It can be used when PGSR.IDONE=1, to manually clear the PGSR status bits, although starting a new init process will automatically clear the PGSR status bits. Or it can be used to manually clear the DXnGSR status bits, although starting a new data training process will automatically clear the DXnGSR status bits.</p>
27:19	RO	0x0	reserved
18	RW	0x0	<p>CTLDINIT Controller DRAM Initialization</p> <p>Indicates if set that DRAM initialization will be performed by the controller. Otherwise if not set it indicates that DRAM initialization will be performed using the built-in initialization sequence or using software through the configuration port.</p>
17	RW	0x0	<p>DLLBYP DLL Bypass</p> <p>A setting of 1 on this bit will put all PHY DLLs in bypass mode. A bypassed DLL is also powered down (disabled).</p>
16	RW	0x0	<p>ICPC Initialization Complete Pin Configuration</p> <p>Specifies how the DFI 2.1 initialization complete output pin should be used to indicate the status of initialization. Valid value are:</p> <p>1'b0: Asserted after PHY initialization (DLL locking and impedance calibration) is complete.</p> <p>1'b1: Asserted after PHY initialization is complete and the triggered the PUBL initialization (DRAM initialization, data training, or initialization trigger with no selected initialization) is complete.</p>
15:9	RO	0x0	reserved
8	RW	0x0	<p>RVTRN Read Valid Training</p> <p>Executes a PUB training routine to determine the optimum position of the read valid signal for maximum system timing margins.</p>

Bit	Attr	Reset Value	Description
7	RW	0x0	QSTRN Read DQS Training Executes a PUBL training routine to determine the optimum position of the read data DQS strobe for maximum system timing margins.
6	RW	0x0	DRAMINIT DRAM Initialization Executes the DRAM initialization sequence
5	RW	0x0	DRAMRST DRAM Reset (DDR3 Only) Issues a reset to the DRAM (by driving the DRAM reset pin low) and wait 200us. This can be triggered in isolation or with the full DRAM initialization (DRAMINIT). For the later case, the reset is issued and 200us is waited before starting the full initialization sequence.
4	RW	0x0	ITMSRST Interface Timing Module Soft Reset Soft resets the interface timing modules for the data and data strobes, i.e., it asserts the ITM soft reset (srstb) signal.
3	RW	0x0	ZCAL Impedance Calibrate Performs PHY impedance calibration
2	RW	0x0	DLLLOCK DLL Lock Waits for the PHY DLLs to lock
1	RW	0x0	DLLSRST DLL Soft Rest Soft resets all PHY DLLs by driving the DLL soft reset pin
0	RW	0x0	INIT Initialization Trigger A write of '1' to this bit triggers the DDR system initialization, including PHY initialization, DRAM initialization, and PHY training. The exact initialization steps to be executed are specified in bits 1 to 6 of this register. A bit setting of 1 means the step will be executed as part of the initialization sequence, while a setting of 0 means the step will be bypassed. The initialization trigger bit is self-clearing.

**DDR\_PUBL\_PGCR**

Address: Operational Base + offset (0x0008)

## PHY General Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	<p>LBMODE Loopback Mode Indicates if set that the PHY/PUB is in loopback mode</p>
30	RW	0x0	<p>LBDQSS Loopback DQS Gating Selects the DQS gating mode that should be used when the PHY is in loopback mode, including BIST loopback mode. Valid values are: 1'b0: DQS gate training will be triggered on the PUB 1'b1: DQS gate is set manually using software</p>
29	RW	0x0	<p>LBDQSS Loopback DQS Shift Selects how the read DQS is shifted during loopback to ensure that the read DQS is centered into the read data eye. Valid values are: 1'b0: PUB sets the read DQS delay to 0; DQS is already shifted 90 degrees by write path 1'b1: The read DQS shift is set manually through software</p>
28:25	RW	0x0	<p>RFSHDT Refresh During Training A non-zero value specifies that a burst of refreshes equal to the number specified in this field should be sent to the SDRAM after training each rank except the last rank.</p>
24	RW	0x1	<p>PDDISDX Power Down Disabled Byte Indicates if set that the DLL and I/Os of a disabled byte should be powered down</p>
23:22	RW	0x2	<p>ZCKSEL Impedance Clock Divider Select Selects the divide ratio for the clock used by the impedance control logic relative to the clock used by the memory controller and SDRAM. Valid values are: 2'b00: Divide by 2 2'b01: Divide by 8 2'b10: Divide by 32 2'b11: Divide by 64</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
21:18	RW	0xf	RANKEN Rank Enable Specifies the ranks that are enabled for data-training. Bit 0 controls rank 0, bit 1 controls rank 1, bit 2 controls rank 2, and bit 3 controls rank 3. Setting the bit to '0' enables the rank, and setting it to '1' disables the rank.
17:16	RW	0x0	IODDRM I/O DDR Mode (D3F I/O Only) Selects the DDR mode for the I/Os
15	RW	0x0	IOLB I/O Loop-Back Select Selects where inside the I/O the loop-back of signals happens. Valid values are: 1'b0: Loopback is after output buffer; output enable must be asserted 1'b1: Loopback is before output buffer; output enable is don't care
14	RW	0x0	CKINV CK Invert Specifies if set that CK/CK# should be inverted. Otherwise CK/CK# toggles with normal polarity.
13:12	RW	0x2	CKDV CK Disable Value Specifies the static value that should be driven on CK/CK# pair(s) when the pair(s) is disabled. CKDV[0] specifies the value for CK and CKDV[1] specifies the value for CK#.
11:9	RW	0x7	CKEN CK Enable Controls whether the CK going to the SDRAM is enabled (toggling) or disabled (static value defined by CKDV). One bit for each of the three CK pairs.

Bit	Attr	Reset Value	Description
8:5	RW	0x0	<p>DTOSEL Digital Test Output Select Selects the PHY digital test output that should be driven onto PHY digital test output (phy_dto) pin: Valid values are:</p> <ul style="list-style-type: none"> <li>4'b0000: DATX8 0 DLL digital test output</li> <li>4'b0001: DATX8 1 DLL digital test output</li> <li>4'b0010: DATX8 2 DLL digital test output</li> <li>4'b0011: DATX8 3 DLL digital test output</li> <li>4'b0100: DATX8 4 DLL digital test output</li> <li>4'b0101: DATX8 5 DLL digital test output</li> <li>4'b0110: DATX8 6 DLL digital test output</li> <li>4'b0111: DATX8 7 DLL digital test output</li> <li>4'b1000: DATX8 8 DLL digital test output</li> <li>4'b1001: AC DLL digital test output</li> <li>4'b1010, 4'b01111: Reserved</li> </ul>
4:3	RW	0x0	<p>DFTLMT DQS Drift Limit Specifies the expected limit of drift on read data strobes. A drift of this value or greater is reported as a drift error through the host port error flag. Valid values are:</p> <ul style="list-style-type: none"> <li>2'b00: No limit (no error reported)</li> <li>2'b01: 90 deg drift</li> <li>2'b10: 180 deg drift</li> <li>2'b11: 270 deg or more drift</li> </ul> <p><i>Note: Although reported through the error flag, this is not an error requiring any action. It is simply an indicator that the drift is greater than expected.</i></p>
2	RW	0x1	<p>DFTCMP DQS Drift Compensation Enables or disables DQS drift compensation. Valid values are:</p> <ul style="list-style-type: none"> <li>1'b0: Disables data strobe drift compensation</li> <li>1'b1: Enables data strobe drift compensation</li> </ul> <p>By default, drift compensation is enabled.</p> <p><i>Note: Drift compensation must be disabled for LPDDR2.</i></p>
1	RW	0x0	<p>DQSCFG DQS Gating Configuration Selects one of the two DQS gating schemes:</p> <ul style="list-style-type: none"> <li>1'b0: DQS gating is shut off using the rising edge of DQS_b (active windowing mode)</li> <li>1'b1: DQS gating blankets the whole burst (passive windowing mode)</li> </ul> <p><i>Note: Passive windowing must be used for LPDDR2.</i></p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	<p>ITMDMD ITM DDR Mode Selects whether ITMS uses DQS and DQS# or it only uses DQS. Valid values are: 1'b0: ITMS uses DQS and DQS# 1'b1: ITMS uses DQS only <i>Note: The only valid value for DDR is 1</i></p>

**DDR\_PUBL\_PGSR**

Address: Operational Base + offset (0x000c)

PHY General Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	<p>TQ Temperature Output (LPDDR Only) Connected to the DRAM TQ pin which is defined to go high when the LPDDR device temperature equals to or exceeds 85C, otherwise it is low.</p>
30:10	RO	0x0	reserved
9	RO	0x0	<p>RVEIRR Read Valid Training Intermittent Error If set, indicates that there was an intermittent error during read valid training, such as a pass was followed by a fail then followed by another pass.</p>
8	RO	0x0	<p>RVERR Read Valid Training Error If set, indicates that a valid read valid placement could not be found during read valid training.</p>
7	RO	0x0	<p>DFTERR DQS Drift Error If set, indicates that at least one of the read data strobes has drifted by more than or equal to the drift limit set in the PHY General Configuration Register (PGCR).</p>
6	RO	0x0	<p>DTIERR Data Training Intermittent Error If set, indicates that there was an intermittent error during data training, such as a pass was followed by a fail then followed by another pass.</p>
5	RO	0x0	<p>DTERR Data Training Error If set, indicates that a valid DQS gating window could not be found during data training.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RO	0x0	DTDONE Data Training Done Indicates, if set, that the PHY has finished doing data training
3	RO	0x0	DIDONE DRAM Initialization Done Indicates if set that DRAM initialization has completed
2	RO	0x0	ZCDONE Impedance Calibration Done Indicates if set that impedance calibration has completed
1	RO	0x0	DLDONE DLL Lock Done Indicates if set that DLL locking has completed
0	RO	0x0	IDONE Initialization Done Indicates if set that the DDR system initialization has completed. This bit is set after all the selected initialization routines in PIR register have completed.

**DDR\_PUBL\_DLLGCR**

Address: Operational Base + offset (0x0010)

DLL General Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved
29	RW	0x0	LOCKDET Master lock detector enable.
28	RO	0x0	reserved
27:20	RW	0x37	SBIAS Slave Bias Trim Used to trim the bias for the slave DLL
19:12	RW	0x37	MBIAS Master Bias Trim Used to trim the bias for the master DLL
11	RW	0x0	TESTSW Test Switch Selects the test signals of either the master DLL ('0') or the slave DLL ('1')

Bit	Attr	Reset Value	Description
10:9	RW	0x0	<p>ATC Analog Test Control Selects the analog signal to be output on the DLL analog test output (test_out_a) when TESTEN is high (Output is Vss when TESTEN is low). The test output either comes from the master DLL or the slave DLL, depending on the setting of the test switch (TESTSW). Both master DLL and slave DLL output similar analog test signals. Valid settings for analog test control are:</p> <ul style="list-style-type: none"> <li>2'b00: Filter output (Vc)</li> <li>2'b01: Replica bias output for NMOS (Vbn)</li> <li>2'b10: Replica bias output for PMOS (Vbp)</li> <li>2'b11: Vdd 00</li> </ul>
8:6	RW	0x0	<p>DTC Digital Test Control Selects the digital signal to be output on the DLL digital test output (test_out_d[1]) when TESTEN is high (Output is '0' when TESTEN is low). Valid settings for master DLL (such as, when TESTSW = '0'):</p> <ul style="list-style-type: none"> <li>3'b000: 0 output clock (clk_0)</li> <li>3'b001: 90 output clock (clk_90)</li> <li>3'b010: 180 output clock (clk_180)</li> <li>3'b011: 270 output clock (clk_270)</li> <li>3'b100: 360 internal clock (clk_360_int)</li> <li>3'b101: Speed-up pulse (spdup)</li> <li>3'b110: Slow-down pulse (slwdn)</li> <li>3'b111: 0 MCTL logic clock (cclk_0)</li> </ul> <p>Valid settings for slave DLL (such as when TESTSW = '1'):</p> <ul style="list-style-type: none"> <li>3'b000: Input DQS strobe (dqs)</li> <li>3'b001: Input clock reference (clk_90_in)</li> <li>3'b010: Internal feedback clock (clk_0_out)</li> <li>3'b011: 90 output DQS_b strobe (dqsb_90)</li> <li>3'b100: 90 output DQS strobe (dqs_90)</li> <li>3'b101: Speed-up pulse (spdup)</li> <li>3'b110: Slow-down pulse (slwdn)</li> <li>3'b111: Auto-lock enable signal</li> </ul>
5	RW	0x0	<p>TESTEN Test Enable Enables digital and analog test outputs selected by DTC and ATC respectively</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:2	RW	0x0	IPUMP Charge Pump Current Trim Used to trim charge pump current: 3'b000: maximum current 3'b111: minimum current
1:0	RW	0x0	DRES Delta Resistor Trim Used to trim reference current versus resistor value variation: 2'b00: Rnom 2'b01: Rnom - 20% 2'b1x: Rnom + 20%

**DDR\_PUBL\_ACDLLCR**

Address: Operational Base + offset (0x0014)

AC DLL Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	DLLDIS DLL Disable A disabled DLL is bypassed. Default ('0') is DLL enabled
30	RW	0x1	DLLSRST DLL Soft Rest Soft resets the AC DLL by driving the DLL soft reset pin
29:19	RO	0x0	reserved
18	RW	0x0	ATESTEN Analog Test Enable Enables the analog test signal to be output on the DLL analog test output (test_out_a). The DLL analog test output is tri-stated when this bit is '0'.
17:12	RO	0x0	Reserved
11:9	RW	0x0	MFWDLY Master Feed-Forward Delay Trim Used to trim the delay in the master DLL feed-forward path: 3'b000: minimum delay 3'b111: maximum delay

Bit	Attr	Reset Value	Description
8:6	RW	0x0	MFBDLY Master Feed-Back Delay Trim Used to trim the delay in the master DLL feedback path: 3'b000: minimum delay 3'b111: maximum delay
5:0	RO	0x0	Reserved

**DDR\_PUBL\_PTR0**

Address: Operational Base + offset (0x0018)

PHY Timing Register 0

Bit	Attr	Reset Value	Description
31:22	RO	0x0	reserved
21:18	RW	0x8	tITMSRST ITM Soft Reset Time Number of controller clock cycles that the ITM soft reset pin must remain asserted when the soft reset is applied to the ITMs. This must correspond to a value that is equal to or more than 8 controller clock cycles. Default value corresponds to 8 controller clock cycles.
17:6	RW	0xabe	tDLLLOCK DLL Lock Time Number of clock cycles for the DLL to stabilize and lock, i.e. number of clock cycles from when the DLL reset pin is de-asserted to when the DLL has locked and is ready for use. Refer to the PHY databook for the DLL lock time. Default value corresponds to 5.12us at 533MHz.
5:0	RW	0x1b	tDLLSRST DLL Soft Reset Time Number of controller clock cycles that the DLL soft reset pin must remain asserted when the soft reset is triggered through the PHY Initialization Register (PIR). This must correspond to a value that is equal to or more than 50ns or 8 controller clock cycles, whichever is bigger. Default value corresponds to 50ns at 533MHz.

**DDR\_PUBL\_PTR1**

Address: Operational Base + offset (0x001c)

PHY Timing Register 1

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:19	RW	0xc0	tDINIT1 DRAM Initialization Time 1 DRAM initialization time corresponding to the following: DDR3: CKE high time to first command (tRFC +10 ns or 5 tCK, whichever value is larger) DDR2: CKE high time to first command (400 ns) DDR: CKE high time to first command (400 ns or 1 tCK) LPDDR2: CKE low time with power and clock stable (100 ns) Default value corresponds to DDR3 360ns at 533MHz.
18:0	RW	0x4111d	tDINIT0 DRAM Initialization Time 0 DRAM initialization time corresponding to the following: DDR3: CKE low time with power and clock stable (500 us) DDR2: CKE low time with power and clock stable (200 us) DDR: CKE low time with power and clock stable (200 us) LPDDR: CKE high time to first command (200 us) LPDDR2: CKE high time to first command (200 us) Default value corresponds to DDR3 500 us at 533MHz.

**DDR\_PUBL\_PTR2**

Address: Operational Base + offset (0x0020)

PHY Timing Register 2

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:17	RW	0x216	tDINIT3 DRAM Initialization Time 3 DRAM initialization time corresponding to the following: LPDDR2: Time from ZQ initialization command to first command (1 us) Default value corresponds to the LPDDR2 1 us at 533MHz.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16:0	RW	0x1a072	tDINIT2 DRAM Initialization Time 2 DRAM initialization time corresponding to the following: DDR3: Reset low time (200 us on power-up or 100 ns after power-up) LPDDR2: Time from reset command to end of auto initialization (1 us + 10 us = 11 us) Default value corresponds to DDR3 200 us at 533MHz.

**DDR\_PUBL\_ACIOCR**

Address: Operational Base + offset (0x0024)

AC I/O Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RW	0x0	ACSR Address/Command Slew Rate (D3F I/O Only) Selects slew rate of the I/O for all address and command pins, as well as the optional DIMM PAR_IN pin and LPDDR TPD pin.
29	RW	0x1	RSTIOM SDRAM Reset I/O Mode Selects SSTL mode (when set to 0) or CMOS mode (when set to 1) of the I/O for SDRAM Reset
28	RW	0x1	RSTPDR SDRAM Reset Power Down Receiver Powers down, when set, the input receiver on the I/O for SDRAM RST# pin
27	RW	0x0	RSTPDD SDRAM Reset Power Down Driver Powers down, when set, the output driver on the I/O for SDRAM RST# pin
26	RW	0x0	RSTODT SDRAM Reset On-Die Termination Enables, when set, the on-die termination on the I/O for SDRAM RST# pin

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
25:22	RW	0xf	<p><b>RANKPDR</b>  Rank Power Down Receiver  Powers down, when set, the input receiver on the I/O CKE[3:0], ODT[3:0], and CS#[3:0] pins.  RANKPDR[0] controls the power down for CKE[0], ODT[0], and CS#[0], RANKPDR[1] controls the power down for CKE[1], ODT[1], and CS#[1], and so on.</p>
21:18	RW	0x0	<p><b>CSPDD</b>  CS# Power Down Driver  Powers down, when set, the output driver on the I/O for CS#[3:0] pins. PDD[0] controls the power down for CS#[0], PDD[1] controls the power down for CS#[1], and so on. CKE and ODT driver power down is controlled by DSGCR register.</p>
17:14	RW	0x0	<p><b>RANKODT</b>  Rank On-Die Termination  Enables, when set, the on-die termination on the I/O for CKE[3:0], ODT[3:0], and CS#[3:0] pins.  RANKODT[0] controls the on-die termination for CKE[0], ODT[0], and CS#[0], RANKODT[1] controls the on-die termination for CKE[1], ODT[1], and CS#[1], and so on.</p>
13:11	RW	0x7	<p><b>CKPDR</b>  CK Power Down Receiver  Powers down, when set, the input receiver on the I/O for CK[0], CK[1], and CK[2] pins, respectively</p>
10:8	RW	0x0	<p><b>CKPDD</b>  CK Power Down Driver  Powers down, when set, the output driver on the I/O for CK[0], CK[1], and CK[2] pins, respectively</p>
7:5	RW	0x0	<p><b>CKODT</b>  CK On-Die Termination  Enables, when set, the on-die termination on the I/O for CK[0], CK[1], and CK[2] pins, respectively</p>
4	RW	0x1	<p><b>ACPDR</b>  AC Power Down Receiver  Powers down, when set, the input receiver on the I/O for RAS#, CAS#, WE#, BA[2:0], and A[15:0] pins, as well as the optional DIMM PAR_IN pin and LPDDR TPD pin.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	ACPDD AC Power Down Driver Powers down, when set, the output driver on the I/O for RAS#, CAS#, WE#, BA[2:0], and A[15:0] pins, as well as the optional DIMM PAR_IN pin and LPDDR TPD pin.
2	RW	0x0	ACODT Address/Command On-Die Termination Enables, when set, the on-die termination on the I/O for RAS#, CAS#, WE#, BA[2:0], and A[15:0] pins, as well as the optional DIMM PAR_IN pin and LPDDR TPD pin.
1	RW	0x1	ACOE Address/Command Output Enable Enables, when set, the output driver on the I/O for all address and command pins, as well as the optional DIMM PAR_IN pin and LPDDR TPD pin.
0	RW	0x0	ACIOM Address/Command I/O Mode Selects SSTL mode (when set to 0) or CMOS mode (when set to 1) of the I/O for all address and command pins, as well as the optional DIMM PAR_IN pin and LPDDR TPD pin.

**DDR\_PUBL\_DXCCR**

Address: Operational Base + offset (0x0028)

DATX8 Common Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved
16	RW	0x0	AWDT Active Window Data Train Indicates if set that data training (DQS gate training and read valid training) should be performed with active DQS gate window. This is just for debug purposes. The default is to perform training with passive windowing.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RW	0x0	<p>RVSEL ITMD Read Valid Select Selects the scheme used for ITMD read valid. Valid values are: 1'b0: ITMD read valid signal is generated by delayed DFI read enable signal. 1'b1: ITMD read valid is generated by the ITMD itseld using asynchronous crossing.</p>
14	RO	0x0	reserved
13:12	RW	0x0	<p>DXSR Data Slew Rate (D3F I/O Only) Selects slew rate of the I/O for DQ, DM, and DQS/DQS# pins of all DATX8 macros.</p>
11:8	RW	0x8	<p>DQSNRES DQS# Resistor Selects the on-die pull-up/pull-down resistor for DQS# pins. Same encoding as DQSRES. <i>Note: DQS# resistor must be connected for LPDDR2</i></p>
7:4	RW	0x0	<p>DQSRES DQS Resistor Selects the on-die pull-down/pull-up resistor for DQS pins. DQSRES[3] selects pull-down (when set to 0) or pull-up (when set to 1). DQSRES[2:0] selects the resistor value as follows: 3'b000: Open: On-die resistor disconnected 3'b001: 688 ohms 3'b010: 611 ohms 3'b011: 550 ohms 3'b100: 500 ohms 3'b101: 458 ohms 3'b110: 393 ohms 3'b111: 344 ohms <i>Note: DQS resistor must be connected for LPDDR2</i></p>
3	RW	0x0	<p>DXPDR Data Power Down Receiver Powers down, when set, the input receiver on I/O for DQ, DM, and DQS/DQS# pins of all DATX8 macros. This bit is ORed with the PDR configuration bit of the individual DATX8.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x0	DXPDD Data Power Down Driver Powers down, when set, the output driver on I/O for DQ, DM, and DQS/DQS# pins of all DATX8 macros. This bit is ORed with the PDD configuration bit of the individual DATX8.
1	RW	0x0	DXIOM Data I/O Mode Selects SSTL mode (when set to 0) or CMOS mode (when set to 1) of the I/O for DQ, DM, and DQS/DQS# pins of all DATX8 macros. This bit is ORed with the IOM configuration bit of the individual DATX8.
0	RW	0x0	DXODT Data On-Die Termination Enables, when set, the on-die termination on the I/O for DQ, DM, and DQS/DQS# pins of all DATX8 macros. This bit is ORed with the ODT configuration bit of the individual DATX8.

**DDR\_PUBL\_DSGCR**

Address: Operational Base + offset (0x002c)

DDR System General Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x1	CKEOE SDRAM CKE Output Enable Enables, when set, the output driver on the I/O for SDRAM CKE pins
30	RW	0x1	RSTOE SDRAM Reset Output Enable Enables, when set, the output driver on the I/O for SDRAM RST# pin
29	RW	0x1	ODTOE SDRAM ODT Output Enable Enables, when set, the output driver on the I/O for SDRAM ODT pins
28	RW	0x1	CKOE SDRAM CK Output Enable Enables, when set, the output driver on the I/O for SDRAM CK/CK# pins

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27	RW	0x1	TPDOE SDRAM TPD Output Enable (LPDDR Only) Enables, when set, the output driver on the I/O for SDRAM TPD pin
26	RW	0x0	TPDPD DRAM TPD Power Down Driver (LPDDR Only) Powers down, when set, the output driver on the I/O for SDRAM TPD pin. Note that the power down of the receiver on the I/O for SDRAM TPD pin is controlled by ACIOCR[ACPDR] register bit.
25	RW	0x1	NL2OE Non-LPDDR2 Output Enable Enables, when set, the output driver on the I/O for non-LPDDR2 (ODT, RAS#, CAS#, WE#, and BA) pins. This may be used when a chip that is designed for both LPDDR2 and other DDR modes is being used in LPDDR2 mode. For these pins, the I/O output enable signal (OE) is an AND of this bit and the respective output enable bit in ACIOCR or DSGCR registers.
24	RW	0x0	NL2PD Non-LPDDR2 Power Down Powers down, when set, the output driver and the input receiver on the I/O for non-LPDDR2 (ODT, RAS#, CAS#, WE#, and BA) pins. This may be used when a chip that is designed for both LPDDR2 and other DDR modes is being used in LPDDR2 mode. For these pins, the I/O power down signal (PDD or PDR) is an OR of this bit and the respective power-down bit in ACIOCR register.
23:20	RW	0x0	ODTPDD ODT Power Down Driver Powers down, when set, the output driver on the I/O for ODT[3:0] pins. ODTPDD[0] controls the power down for ODT[0], ODTPDD[1] controls the power down for ODT[1], and so on.
19:16	RW	0x0	CKEPDD CKE Power Down Driver Powers down, when set, the output driver on the I/O for CKE[3:0] pins. CKEPDD[0] controls the power down for CKE[0], CKEPDD[1] controls the power down for CKE[1], and so on.
15:13	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x0	<p><b>FXDLAT</b> Fixed Latency Specified whether all reads should be returned to the controller with a fixed read latency. Enabling fixed read latency increases the read latency. Valid values are: 1'b0: Disable fixed read latency 1'b1: Enable fixed read latency If the design is compiled for HDR mode, then either NOBUB or FXDLAT must be set to 1.</p>
11	RW	0x0	<p><b>NOBUB</b> No Bubbles Specified whether reads should be returned to the controller with no bubbles. Enabling no-bubble reads increases the read latency. Valid values are: 1'b0: Bubbles are allowed during reads 1'b1: Bubbles are not allowed during reads If the design is compiled for HDR mode, then either NOBUB or FXDLAT must be set to 1.</p>
10:8	RW	0x0	<p><b>DQSGE</b> DQS Gate Early Specifies the number of clock cycles for which the DQS gating must be enabled earlier than its normal position. Only applicable when using PDQSR I/O cell, passive DQS gating and no drift compensation. This field is recommended to be set to zero for all DDR types other than LPDDR2. For LPDDR2 it should be set to (tDQSCKmax - tDQSCK) divide by clock period and rounded up.</p>
7:5	RW	0x0	<p><b>DQSGX</b> DQS Gate Extension Specifies the number of clock cycles for which the DQS gating must be extended beyond the normal burst length width. Only applicable when using PDQSR I/O cell, passive DQS gating and no drift compensation. This field is recommended to be set to zero for all DDR types other than LPDDR2. For LPDDR2 it should be set to (tDQSCKmax - tDQSCK) divide by clock period and rounded up.</p>
4	RW	0x1	<p><b>LPDLLPD</b> Low Power DLL Power Down Specifies if set that the PHY should respond to the DFI low power opportunity request and power down the DLL of the byte if the wakeup time request satisfies the DLL lock time.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x1	LPIOPD Low Power I/O Power Down Specifies if set that the PHY should respond to the DFI low power opportunity request and power down the I/Os of the byte.
2	RW	0x1	ZUEN Impedance Update Enable Specifies if set that the PHY should perform impedance calibration (update) whenever there is a controller initiated DFI update request. Otherwise the PHY will ignore an update request from the controller.
1	RW	0x1	BDISEN Byte Disable Enable Specifies if set that the PHY should respond to DFI byte disable request. Otherwise the byte disable from the DFI is ignored in which case bytes can only be disabled using the DXnGCR register.
0	RW	0x1	PUREN PHY Update Request Enable Specifies if set, that the PHY should issue PHY-initiated DFI update request when there is DQS drift of more than 3/4 of a clock cycle within one continuous (back-to-back) read burst. By default the PHY issues PHY-initiated update requests and the controller should respond otherwise the PHY may return erroneous values. The option to disable it is provided only for silicon evaluation and testing.

**DDR\_PUBL\_DCR**

Address: Operational Base + offset (0x0030)

DRAM Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	TPD Test Power Down (LPDDR Only) If set will place the DRAM in deep power down mode
30:29	RO	0x0	reserved
28	RW	0x0	DDR2T DDR 2T Timing Indicates if set that 2T timing should be used by PUB internally generated SDRAM transactions.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27	RW	0x0	NOSRA No Simultaneous Rank Access Specifies if set that simultaneous rank access on the same clock cycle is not allowed. This means that multiple chip select signals should not be asserted at the same time. This may be required on some DIMM systems.
26:10	RO	0x0	reserved
9:8	RW	0x0	DDRTYPE DDR Type Selects the DDR type for the specified DDR mode. Valid values for LPDDR2 are: 2'b00: LPDDR2-S4 2'b01: LPDDR2-S2 2'b10: LPDDR2-NVM 2'b11: Reserved
7	RW	0x0	MPRDQ Multi-Purpose Register (MPR) DQ (DDR3 Only) Specifies the value that is driven on non-primary DQ pins during MPR reads. Valid values are: 1'b0: Primary DQ drives out the data from MPR (0-1-0-1); non-primary DQs drive '0' 1'b1: Primary DQ and non-primary DQs all drive the same data from MPR (0-1-0-1)
6:4	RW	0x0	PDQ Primary DQ (DDR3 Only) Specifies the DQ pin in a byte that is designated as a primary pin for Multi-Purpose Register (MPR) reads. Valid values are 0 to 7 for DQ[0] to DQ[7], respectively.
3	RW	0x1	DDR8BNK DDR 8-Bank Indicates if set that the SDRAM used has 8 banks. tRPA = tRP+1 and tFAW are used for 8-bank DRAMs, other tRPA = tRP and no tFAW is used. Note that a setting of 1 for DRAMs that have fewer than 8 banks still results in correct functionality but less tighter DRAM command spacing for the parameters described here.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:0	RW	0x3	DDRMD DDR Mode SDRAM DDR mode. Valid values are: 3'b000: LPDDR (Mobile DDR) 3'b001: DDR 3'b010: DDR2 3'b011: DDR3 3'b100: LPDDR2 (Mobile DDR2) 3'b101, 3'b111: Reserved

**DDR\_PUBL\_DTPRO**

Address: Operational Base + offset (0x0034)

DRAM Timing Parameters Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	tCCD Read to read and write to write command delay. Valid values are: 1'b0: BL/2 for DDR2 and 4 for DDR3 1'b1: BL/2 + 1 for DDR2 and 5 for DDR3
30:25	RW	0x18	tRC Activate to activate command delay (same bank). Valid values are 2 to 42.
24:21	RW	0x4	tRRD Activate to activate command delay (different banks). Valid values are 1 to 8.
20:16	RW	0x12	tRAS Activate to precharge command delay. Valid values are 2 to 31.
15:12	RW	0x6	tRCD Activate to read or write delay. Minimum time from when an activate command is issued to when a read or write to the activated row can be issued. Valid values are 2 to 11.
11:8	RW	0x6	tRP Precharge command period The minimum time between a precharge command and any other command. Note that the Controller automatically derives tRPA for 8-bank DDR2 devices by adding 1 to tRP . Valid values are 2 to 11.
7:5	RW	0x3	tWTR Internal write to read command delay. Valid values are 1 to 6.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:2	RW	0x3	tRTP Internal read to precharge command delay. Valid values are 2 to 6. Note that even though RTP does not apply to JEDEC DDR devices, this parameter must still be set to a minimum value of 2 for DDR because the Controller always uses the DDR2 equation, AL + BL/2 + max(RTP,2) - 2, to compute the read to precharge timing (which is BL/2 for JEDEC DDR).
1:0	RW	0x2	tMRD Load mode cycle time The minimum time between a load mode register command and any other command. For DDR3 this is the minimum time between two load mode register commands. Valid values for DDR2 are 2 to 3. For DDR3, the value used for tMRD is 4 plus the value programmed in these bits, i.e. tMRD value for DDR3 ranges from 4 to 7.

**DDR\_PUBL\_DTPR1**

Address: Operational Base + offset (0x0038)

DRAM Timing Parameters Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved
29:27	RW	0x1	tDQSCKmax Maximum DQS output access time from CK/CK# (LPDDR2 only). This value is used for implementing read-to-write spacing. Valid values are 1 to 7.
26:24	RW	0x1	tDQSCK DQS output access time from CK/CK# (LPDDR2 only). This value is used for computing the read latency. Valid values are 1 to 7.. This value is derived from the corresponding parameter in the SDRAM datasheet divided by the clock cycle time without rounding up. The fractional remainder is automatically adjusted for by data training in quarter clock cycle units. If data training is not performed then this fractional remainder must be converted to quarter clock cycle units and the gating registers (DXnDQSTR) adjusted accordingly.

Bit	Attr	Reset Value	Description
23:16	RW	0x83	tRFC Refresh-to-Refresh: Indicates the minimum time, in clock cycles, between two refresh commands or between a refresh and an active command. This is derived from the minimum refresh interval from the datasheet, tRFC(min), divided by the clock cycle time. The default number of clock cycles is for the largest JEDEC tRFC(min) parameter value supported.
15:12	RO	0x0	reserved
11	RW	0x0	tRTODT Read to ODT delay (DDR3 only). Specifies whether ODT can be enabled immediately after the read post-amble or one clock delay has to be added. Valid values are: 1'b0: ODT may be turned on immediately after read post-amble 1'b1: ODT may not be turned on until one clock after the read post-amble If tRTODT is set to 1, then the read-to-write latency is increased by 1 if ODT is enabled.
10:9	RW	0x0	tMOD Load mode update delay (DDR3 only). The minimum time between a load mode register command and a non-load mode register command. Valid values are: 2'b00: 12 2'b01: 13 2'b10: 14 2'b11: 15
8:3	RW	0x12	tFAW 4-bank activate period. No more than 4-bank activate commands may be issued in a given tFAW period. Only applies to 8-bank devices. Valid values are 2 to 31.
2	RW	0x0	tRTW Read to Write command delay. Valid values are: 1'b0: standard bus turn around delay 1'b1: add 1 clock to standard bus turn around delay This parameter allows the user to increase the delay between issuing Write commands to the SDRAM when preceded by Read commands. This provides an option to increase bus turn-around margin for high frequency systems.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RW	0x0	<p>tAOND_tAOFD ODT turn-on/turn-off delays (DDR2 only). The delays are in clock cycles. Valid values are: 2'b00: 2/2.5 2'b01: 3/3.5 2'b10: 4/4.5 2'b11: 5/5.5</p> <p>Most DDR2 devices utilize a fixed value of 2/2.5. For non-standard SDRAMs, the user must ensure that the operational Write Latency is always greater than or equal to the ODT turn-on delay. For example, a DDR2 SDRAM with CAS latency set to 3 and CAS additive latency set to 0 has a Write Latency of 2. Thus 2/2.5 can be used, but not 3/3.5 or higher.</p>

**DDR\_PUBL\_DTPR2**

Address: Operational Base + offset (0x003c)

DRAM Timing Parameters Register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:19	RW	0x200	tDLLK DLL locking time. Valid values are 2 to 1023.
18:15	RW	0x3	tCKE CKE minimum pulse width. Also specifies the minimum time that the SDRAM must remain in power down or self refresh mode. For DDR3 this parameter must be set to the value of tCKESR which is usually bigger than the value of tCKE. Valid values are 2 to 15.
14:10	RW	0x08	tXP Power down exit delay. The minimum time between a power down exit command and any other command. This parameter must be set to the maximum of the various minimum power down exit delay parameters specified in the SDRAM datasheet, i.e. max(tXP , tXARD, tXARDS) for DDR2 and max(tXP , tXP DLL) for DDR3. Valid values are 2 to 31.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9:0	RW	0x0c8	tXS Self refresh exit delay. The minimum time between a self refresh exit command and any other command. This parameter must be set to the maximum of the various minimum self refresh exit delay parameters specified in the SDRAM datasheet, i.e. max(tXSNR, tXSRD) for DDR2 and max(tXS, tXSDL) for DDR3. Valid values are 2 to 1023.

**DDR\_PUBL\_MR0**

Address: Operational Base + offset (0x0040)

Mode Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12	RW	0x0	PD Power-Down Control Controls the exit time for power-down modes. Refer to SDRAM datasheet for details on power-down modes. Valid values are: 1'b0: Slow exit (DLL off) 1'b1: Fast exit (DLL on)
11:9	RW	0x5	WR Write Recovery This is the value of the write recovery in clock cycles. It is calculated by dividing the datasheet write recovery time, tWR (ns) by the datasheet clock cycle time, tCK (ns) and rounding up a non-integer value to the next integer. Valid values are: 3'b001: 5 3'b010: 6 3'b011: 7 3'b100: 8 3'b101: 10 3'b110: 12 All other settings are reserved and should not be used. <i>NOTE: tWR (ns) is the time from the first SDRAM positive clock edge after the last data-in pair of a write command, to when a precharge of the same bank can be issued.</i>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	RW	0x0	DR DLL Reset Writing a '1' to this bit will reset the SDRAM DLL. This bit is self-clearing, i.e. it returns back to '0' after the DLL reset has been issued.
7	RW	0x0	TM Operating Mode Selects either normal operating mode (0) or test mode (1). Test mode is reserved for the manufacturer and should not be used.
6:4	RW	0x5	CL_1 CAS Latency The delay, in clock cycles, between when the SDRAM registers a read command to when data is available. Valid values are: 4'b0010: 5 4'b0100: 6 4'b0110: 7 4'b1000: 8 4'b1010: 9 4'b1100: 10 4'b1110: 11 All other settings are reserved and should not be used.
3	RW	0x0	BT Burst Type Indicates whether a burst is sequential (0) or interleaved (1).
2	RW	0x0	CL_0 CAS Latency merged with bit6-4
1:0	RW	0x2	BL Burst Length Determines the maximum number of column locations that can be accessed during a given read or write command. Valid values for DDR3 are: 2'b00: 8 (Fixed) 2'b01: 4 or 8 (On the fly) 2'b10: 4 (Fixed) 2'b11: Reserved

**DDR\_PUBL\_MR1**

Address: Operational Base + offset (0x0044)

## Mode Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12	RW	0x0	<p>QOFF Output Enable/Disable When '0' all outputs function normal; when '1' all SDRAM outputs are disabled removing output buffer current. This feature is intended to be used for IDD characterization of read current and should not be used in normal operation.</p>
11	RW	0x0	<p>TDQS Termination Data Strobe When enabled ('1') TDQS provides additional termination resistance outputs that may be useful in some system configurations. Refer to the SDRAM datasheet for details.</p>
10	RO	0x0	reserved
9	RW	0x0	<p>RTT_2 On Die Termination Selects the effective resistance for SDRAM on die termination. Valid values are: 3'b000: ODT disabled 3'b001: RZQ/4 3'b010: RZQ/2 3'b011: RZQ/6 3'b100: RZQ/12 3'b101: RZQ/8 All other settings are reserved and should not be used.</p>
8	RO	0x0	reserved
7	RW	0x0	<p>LEVEL Write Leveling Enable Enables write-leveling when set</p>
6	RW	0x0	<p>RTT_1 On Die Termination merged with bit9.</p>
5	RW	0x0	<p>DIC_1 Output Driver Impedance Control Controls the output drive strength. Valid values are: 2'b00: Reserved for RZQ/6 2'b01: RZQ7 2'b10: Reserved 2'b11: Reserved</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:3	RW	0x0	<p>AL Posted CAS Additive Latency Setting additive latency that allows read and write commands to be issued to the SDRAM earlier than normal (refer to SDRAM datasheet for details). Valid values are: 2'b00: 0 (AL disabled) 2'b01: CL - 1 2'b10: CL - 2 2'b11: Reserved</p>
2	RW	0x0	<p>RTT_0 On Die Termination merged with bit9.</p>
1	RW	0x0	<p>DIC_0 Output Driver Impedance Control Controls the output drive strength. Merged with bit5.</p>
0	RW	0x0	<p>DE DLL Enable/Disable Enable (0) or disable (1) the DLL. DLL must be enabled for normal operation.</p>

**DDR\_PUBL\_MR2**

Address: Operational Base + offset (0x0048)

Mode Register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved
10:9	RW	0x0	<p>RTTWR Dynamic ODT Selects RTT for dynamic ODT. Valid values are: 2'b00: Dynamic ODT off 2'b01: RZQ/4 2'b10: RZQ/2 2'b11: Reserved</p>
8	RO	0x0	reserved
7	RW	0x0	<p>SRT Self-Refresh Temperature Range Selects either normal ('0') or extended ('1') operating temperature range during self-refresh.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RW	0x0	<p>ASR Auto Self-Refresh When enabled ('1'), SDRAM automatically provides self-refresh power management functions for all supported operating temperature values. Otherwise the SRT bit must be programmed to indicate the temperature range.</p>
5:3	RW	0x0	<p>CWL CAS Write Latency The delay, in clock cycles, between when the SDRAM registers a write command to when write data is available. Valid values are: 3'b000: 5 (tCK = 2.5ns) 3'b001: 6 (2.5ns &gt; tCK = 1.875ns) 3'b010: 7 (1.875ns &gt; tCK = 1.5ns) 3'b011: 8 (1.5ns &gt; tCK = 1.25ns) All other settings are reserved and should not be used</p>
2:0	RW	0x0	<p>PASR Partial Array Self Refresh Specifies that data located in areas of the array beyond the specified location will be lost if self refresh is entered. Valid settings for 4 banks are: 3'b000: Full Array 3'b001: Half Array (BA[1:0] = 00 &amp; 01) 3'b010: Quarter Array (BA[1:0] = 00) 3'b011: Not defined 3'b100: 3/4 Array (BA[1:0] = 01, 10, &amp; 11) 3'b101: Half Array (BA[1:0] = 10 &amp; 11) 3'b110: Quarter Array (BA[1:0] = 11) 3'b111: Not defined Valid settings for 8 banks are: 3'b000: Full Array 3'b001: Half Array (BA[2:0] = 000, 001, 010 &amp; 011) 3'b010: Quarter Array (BA[2:0] = 000, 001) 3'b011: 1/8 Array (BA[2:0] = 000) 3'b100: 3/4 Array (BA[2:0] = 010, 011, 100, 101, 110 &amp; 111) 3'b101: Half Array (BA[2:0] = 100, 101, 110 &amp; 111) 3'b110: Quarter Array (BA[2:0] = 110 &amp; 111) 3'b111: 1/8 Array (BA[2:0] 111)</p>

**DDR\_PUBL\_MR3**

Address: Operational Base + offset (0x004c)

Mode Register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2	RW	0x0	MPR Multi-Purpose Register Enable Enables, if set, that read data should come from the Multi-Purpose Register. Otherwise read data come from the DRAM array.
1:0	RW	0x0	MPRLOC Multi-Purpose Register (MPR) Location Selects MPR data location: Valid value are: 2'b00: Predefined pattern for system calibration All other settings are reserved and should not be used.

**DDR\_PUBL\_ODTCR**

Address: Operational Base + offset (0x0050)

ODT Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23:20	RW	0x2	WRODT1 Write ODT Specifies whether ODT should be enabled ('1') or disabled ('0') on each of the up to four ranks when a write command is sent to rank 0. WRODT0, WRODT1 specify ODT settings when a write is to rank 0, rank 1 respectively. The four bits of each field each represent a rank, the LSB being rank 0 and the MSB being rank 1. Default is to enable ODT only on rank being written to.
19:16	RW	0x1	WRODT0 Write ODT Specifies whether ODT should be enabled ('1') or disabled ('0') on each of the up to four ranks when a write command is sent to rank 0. WRODT0, WRODT1 specify ODT settings when a write is to rank 0, rank 1 respectively. The four bits of each field each represent a rank, the LSB being rank 0 and the MSB being rank 1. Default is to enable ODT only on rank being written to.
15:8	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:4	RW	0x0	<p>RDODT1 Read ODT Specifies whether ODT should be enabled ('1') or disabled ('0') on each of the up to four ranks when a read command is sent to rank 1. RDODT0, RDODT1 specify ODT settings when a read is to rank 0, and rank 1, respectively. The two bits of each field each represent a rank, the LSB being rank 0 and the MSB being rank 1. Default is to disable ODT during reads.</p>
3:0	RW	0x0	<p>RDODT0 Read ODT Specifies whether ODT should be enabled ('1') or disabled ('0') on each of the up to four ranks when a read command is sent to rank 0. RDODT0, RDODT1 specify ODT settings when a read is to rank 0, and rank 1, respectively. The two bits of each field each represent a rank, the LSB being rank 0 and the MSB being rank 1. Default is to disable ODT during reads.</p>

**DDR\_PUBL\_DTAR**

Address: Operational Base + offset (0x0054)

Data Training Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	<p>DTMPR Data Training Using MPR (DDR3 Only) Specifies, if set, that data-training should use the SDRAM Multi-Purpose Register (MPR) register. Otherwise data-training is performed by first writing to some locations in the SDRAM and then reading them back.</p>
30:28	RW	0x0	<p>DTBANK Data Training Bank Address Selects the SDRAM bank address to be used during data training</p>
27:12	RW	0x0000	<p>DTROW Data Training Row Address Selects the SDRAM row address to be used during data training</p>
11:0	RW	0x000	<p>DTCOL Data Training Column Address Selects the SDRAM column address to be used during data training. The lower four bits of this address must always be "0000"</p>

**DDR\_PUBL\_DTDRO**

Address: Operational Base + offset (0x0058)

Data Training Data Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0xdd	DTBYTE3 Data Training Data The fourth 4 bytes of data used during data training. This same data byte is used for each Byte Lane. Default sequence is a walking 1 while toggling data every data cycle.
23:16	RW	0x22	DTBYTE2 Data Training Data The third 4 bytes of data used during data training. This same data byte is used for each Byte Lane. Default sequence is a walking 1 while toggling data every data cycle.
15:8	RW	0xee	DTBYTE1 Data Training Data The second 4 bytes of data used during data training. This same data byte is used for each Byte Lane. Default sequence is a walking 1 while toggling data every data cycle.
7:0	RW	0x11	DTBYTE0 Data Training Data The first 4 bytes of data used during data training. This same data byte is used for each Byte Lane. Default sequence is a walking 1 while toggling data every data cycle.

**DDR\_PUBL\_DTDRI**

Address: Operational Base + offset (0x005c)

Data Training Data Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x77	DTBYTE7 Data Training Data The eighth 4 bytes of data used during data training. This same data byte is used for each Byte Lane. Default sequence is a walking 1 while toggling data every data cycle.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23:16	RW	0x88	DTBYTE6 Data Training Data The seventh 4 bytes of data used during data training. This same data byte is used for each Byte Lane. Default sequence is a walking 1 while toggling data every data cycle.
15:8	RW	0xbb	DTBYTE5 Data Training Data The sixth 4 bytes of data used during data training. This same data byte is used for each Byte Lane. Default sequence is a walking 1 while toggling data every data cycle.
7:0	RW	0x44	DTBYTE4 Data Training Data The fifth 4 bytes of data used during data training. This same data byte is used for each Byte Lane. Default sequence is a walking 1 while toggling data every data cycle.

**DDR\_PUBL\_DCUAR**

Address: Operational Base + offset (0x00c0)

DCU Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11	RW	0x0	ATYPE Access Type Specifies the type of access to be performed using this address. Valid values are: 1'b0: Write access 1'b1: Read access
10	RW	0x0	INCA Increment Address Specifies, if set, that the cache address specified in WADDR and SADDR should be automatically incremented after each access of the cache. The increment happens in such a way that all the slices of a selected word are first accessed before going to the next word.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9:8	RW	0x0	CSEL Cache Select Selects the cache to be accessed. Valid values are: 2'b00: Command cache 2'b01: Expected data cache 2'b10: Read data cache 2'b11: Reserved
7:4	RW	0x0	CSADDR Cache Slice Address Address of the cache slice to be accessed
3:0	RW	0x0	CWADDR Cache Word Address Address of the cache word to be accessed

**DDR\_PUBL\_DCUDR**

Address: Operational Base + offset (0x00c4)

DCU Data Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	CDATA Cache Data Data to be written to or read from a cache. This data corresponds to the cache word slice specified by the DCU Address Register.

**DDR\_PUBL\_DCURR**

Address: Operational Base + offset (0x00c8)

DCU Run Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23	RW	0x0	XCEN Expected Compare Enable Indicates if set that read data coming back from the SDRAM should be compared with the expected data
22	RW	0x0	RCEN Read Capture Enable Indicates if set that read data coming back from the SDRAM should be captured into the read data cache
21	RW	0x0	SCOF Stop Capture On Full Specifies if set that the capture of read data should stop when the capture cache is full

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
20	RW	0x0	<p><b>SONF</b> Stop On Nth Fail Specifies if set that the execution of commands and the capture of read data should stop when there are N read data failures. The number of failures is specified by NFAIL. Otherwise commands execute until the end of the program or until manually stopped using a STOP command.</p>
19:12	RW	0x00	<p><b>NFAIL</b> Number of Failures Specifies the number of failures after which the execution of commands and the capture of read data should stop if SONF bit of this register is set. Execution of commands and the capture of read data will stop after (NFAIL+1) failures if SONF is set.</p>
11:8	RW	0x0	<p><b>EADDR</b> End Address Cache word address where the execution of command should end</p>
7:4	RW	0x0	<p><b>SADDR</b> Start Address Cache word address where the execution of commands should begin</p>
3:0	RW	0x0	<p><b>DINST</b> DCU Instruction Selects the DCU command to be executed: Valid values are: 4'b0000: NOP: No operation 4'b0001: Run: Triggers the execution of commands in the command cache. 4'b0010: Stop: Stops the execution of commands in the command cache. 4'b0011: Stop Loop: Stops the execution of an infinite loop in the command cache. 4'b0100: Reset: Resets all DCU run time registers. 4'b0101 - 4'b1111: Reserved</p>

**DDR\_PUBL\_DCULR**

Address: Operational Base + offset (0x00cc)  
DCU Loop Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RW	0x0	XLEADDR Expected Data Loop End Address The last expected data cache word address that contains valid expected data. Expected data should looped between 0 and this address.
27:18	RO	0x0	reserved
17	RW	0x0	IDA Increment DRAM Address Indicates if set that DRAM addresses should be incremented every time a DRAM read/write command inside the loop is executed
16	RW	0x0	LINF Loop Infinite Indicates if set that the loop should be executed indefinitely until stopped by the STOP command. Otherwise the loop is execute LCNT times.
15:8	RW	0x00	LCNT Loop Count The number of times that the loop should be executed if LINF is not set
7:4	RW	0x0	LEADDR Loop End Address Command cache word address where the loop should end
3:0	RW	0x0	LSADDR Loop Start Address Command cache word address where the loop should start

**DDR\_PUBL\_DCUGCR**

Address: Operational Base + offset (0x00d0)

DCU General Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	RCSW Read Capture Start Word The capture and compare of read data should start after Nth word. For example setting this value to 12 will skip the first 12 read data.

**DDR\_PUBL\_DCUTPR**

Address: Operational Base + offset (0x00d4)

DCU Timing Parameters Registers

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	tDCUT3 DCU Generic Timing Parameter 3.
23:16	RW	0x00	tDCUT2 DCU Generic Timing Parameter 2.
15:8	RW	0x00	tDCUT1 DCU Generic Timing Parameter 1.
7:0	RW	0x00	tDCUT0 DCU Generic Timing Parameter 0.

**DDR\_PUBL\_DCUSR0**

Address: Operational Base + offset (0x00d8)

DCU Status Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2	RO	0x0	CFULL Capture Full Indicates if set that the capture cache is full
1	RO	0x0	CFAIL Capture Fail Indicates if set that at least one read data word has failed
0	RO	0x0	RDONE Run Done: Indicates if set that the DCU has finished executing the commands in the command cache. This bit is also set to indicate that a STOP command has successfully been executed and command execution has stopped.

**DDR\_PUBL\_DCUSR1**

Address: Operational Base + offset (0x00dc)

DCU Status Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x00	LPCNT Loop Count Indicates the value of the loop count. This is useful when the program has stopped because of failures to assess how many reads were executed before first fail.
23:16	RO	0x00	FLCND Fail Count Number of read words that have failed

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RO	0x0000	RDCNT Read Count Number of read words returned from the SDRAM

**DDR\_PUBL\_BISTR**

Address: Operational Base + offset (0x0100)

BIST Run Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RO	0x0	reserved
25:23	RW	0x0	BCKSEL BIST CK Select Selects the CK to be used for capturing loopback data on the address/command lane. Valid values are: 3'b000: CK[0] 3'b001: CK[1] 3'b010: CK[2] 3'b011: Reserved 3'b100: CK#[0] 3'b101: CK#[1] 3'b110: CK#[2] 3'b111: Reserved
22:19	RW	0x0	BDXSEL BIST DATX8 Select Select the byte lane for comparison of loopback/read data. Valid values are 0 to 8.
18:17	RW	0x0	BDPAT BIST Data Pattern Selects the data pattern used during BIST. Valid values are: 2'b00: Walking 0 2'b01: Walking 1 2'b10: LFSR-based pseudo-random 2'b11: User programmable
16	RW	0x0	BDMEN BIST Data Mask Enable Enables if set that the data mask BIST should be included in the BIST run, i.e. data pattern generated and loopback data compared. This is valid only for loopback mode.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RW	0x0	BACEN BIST AC Enable Enables the running of BIST on the address/command lane PHY. This bit is exclusive with BDXEN, i.e. both cannot be set to '1' at the same time.
14	RW	0x0	BDXEN BIST DATX8 Enable Enables the running of BIST on the data byte lane PHYs. This bit is exclusive with BACEN, i.e. both cannot be set to '1' at the same time.
13	RW	0x0	BSONF BIST Stop On Nth Fail Specifies if set that the BIST should stop when an nth data word or address/command comparison error has been encountered.
12:5	RW	0x00	NFAIL Number of Failures Specifies the number of failures after which the execution of commands and the capture of read data should stop if BSONF bit of this register is set. Execution of commands and the capture of read data will stop after (NFAIL+1) failures if BSONF is set.
4	RW	0x0	BINF BIST Infinite Run Specifies if set that the BIST should be run indefinitely until when it is either stopped or a failure has been encountered. Otherwise BIST is run until number of BIST words specified in the BISTWCR register has been generated.
3	RW	0x0	BMODE BIST Mode Selects the mode in which BIST is run. Valid values are: 1'b0: Loopback mode: Address, commands and data loop back at the PHY I/Os. 1'b1: DRAM mode: Address, commands and data go to DRAM for normal memory accesses.

Bit	Attr	Reset Value	Description
2:0	RW	0x0	<p>BINST BIST Instruction Selects the BIST instruction to be executed: Valid values are: 3'b000: NOP: No operation 3'b001: Run: Triggers the running of the BIST. 3'b010: Stop: Stops the running of the BIST. 3'b011: Reset: Resets all BIST run-time registers, such as error counters. 3'b100 - 3'b111: Reserved</p>

**DDR\_PUBL\_BISTMSKRO**

Address: Operational Base + offset (0x0104)

BIST Mask Register 0

Bit	Attr	Reset Value	Description
31:28	RW	0x0	ODTMSK Mask bit for each of the up to 4 ODT bits
27:24	RW	0x0	CSMSK Mask bit for each of the up to 4 CS# bits
23:20	RW	0x0	CKEMSK Mask bit for each of the up to 4 CKE bits
19	RW	0x0	WEMSK Mask bit for the WE#
18:16	RW	0x0	BAMSK Mask bit for each of the up to 3 bank address bits
15:0	RW	0x0000	AMSK Mask bit for each of the up to 16 address bits

**DDR\_PUBL\_BISTMSKR1**

Address: Operational Base + offset (0x0108)

BIST Mask Register 1

Bit	Attr	Reset Value	Description
31	RW	0x0	TPDMSK Mask bit for the TPD. LPDDR Only.
30	RW	0x0	PARMSK Mask bit for the PAR_IN. Only for DIMM parity support.
29:20	RO	0x0	reserved
19	RW	0x0	CASMSK Mask bit for the CAS
18	RW	0x0	RASMSK Mask bit for the RAS

Bit	Attr	Reset Value	Description
17:16	RW	0x0	DMMSK Mask bit for the data mask (DM) bits
15:0	RW	0x0000	DQMSK Mask bit for each of the 8 data (DQ) bits

**DDR\_PUBL\_BISTWCR**

Address: Operational Base + offset (0x010c)

BIST Word Count Register

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RW	0x0020	BWCNT BIST Word Count Indicates the number of words to generate during BIST. This must be a multiple of DRAM burst length (BL) divided by 2, e.g. for BL=8, valid values are 4, 8, 12, 16, and so on.

**DDR\_PUBL\_BISTLSR**

Address: Operational Base + offset (0x0110)

BIST LFSR Seed Register

Bit	Attr	Reset Value	Description
31:0	RW	0x1234abcd	SEED LFSR seed for pseudo-random BIST patterns.

**DDR\_PUBL\_BISTAR0**

Address: Operational Base + offset (0x0114)

BIST Address Register 0

Bit	Attr	Reset Value	Description
31	RO	0x0	reserved
30:28	RW	0x0	BBANK BIST Bank Address Selects the SDRAM bank address to be used during BIST
27:12	RW	0x0000	BROW BIST Row Address Selects the SDRAM row address to be used during BIST

Bit	Attr	Reset Value	Description
11:0	RW	0x000	BCOL BIST Column Address Selects the SDRAM column address to be used during BIST. The lower bits of this address must be "0000" for BL16, "000" for BL8, "00" for BL4 and "0" for BL2.

**DDR\_PUBL\_BISTAR1**

Address: Operational Base + offset (0x0118)

BIST Address Register 1

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:4	RW	0x000	BAINC BIST Address Increment Selects the value by which the SDRAM address is incremented for each write/read access. This value must be at the beginning of a burst boundary, i.e. the lower bits must be "0000" for BL16, "00" for BL8, "00" for BL4 and "0" for BL2.
3:2	RW	0x3	BMRANK BIST Maximum Rank Specifies the maximum SDRAM rank to be used during BIST. The default value is set to maximum ranks minus 1. Example default shown here is for a 4-rank system
1:0	RW	0x0	BRANK BIST Rank Selects the SDRAM rank to be used during BIST. Valid values range from 0 to maximum ranks minus 1.

**DDR\_PUBL\_BISTAR2**

Address: Operational Base + offset (0x011c)

BIST Address Register 2

Bit	Attr	Reset Value	Description
31	RO	0x0	reserved
30:28	RW	0x7	BMBANK BIST Maximum Bank Address Specifies the maximum SDRAM bank address to be used during BIST before the address increments to the next rank.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27:12	RW	0xffff	BMROW BIST Maximum Row Address Specifies the maximum SDRAM row address to be used during BIST before the address increments to the next bank.
11:0	RW	0xffff	BMCOL BIST Maximum Column Address Specifies the maximum SDRAM column address to be used during BIST before the address increments to the next row.

**DDR\_PUBL\_BISTUDPR**

Address: Operational Base + offset (0x0120)

BIST User Data Pattern Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0xffff	BUDP1 BIST User Data Pattern 1 Data to be applied on odd DQ pins during BIST
15:0	RW	0x0000	BUDP0 BIST User Data Pattern 0 Data to be applied on even DQ pins during BIST

**DDR\_PUBL\_BISTGSR**

Address: Operational Base + offset (0x0124)

BIST General Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	CASBER CAS Bit Error Indicates the number of bit errors on CAS
29:28	RO	0x0	RASBER RAS Bit Error Indicates the number of bit errors on RAS
27:24	RO	0x0	DMBER DM Bit Error Indicates the number of bit errors on data mask (DM) bit. DMBER[1:0] are for the first DM beat, and DMBER[3:2] are for the second DM beat.
23:22	RO	0x0	TPDBER TPD Bit Error (LPDDR Only) Indicates the number of bit errors on TPD

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
21:20	RO	0x0	PARBER PAR_IN Bit Error (DIMM Only) Indicates the number of bit errors on PAR_IN
19:3	RO	0x0	reserved
2	RO	0x0	BDXERR BIST Data Error Indicates if set that there is a data comparison error in the byte lane
1	RO	0x0	BACERR BIST Address/Command Error Indicates if set that there is a data comparison error in the address/command lane
0	RO	0x0	BDONE BIST Done Indicates if set that the BIST has finished executing. This bit is reset to zero when BIST is triggered.

**DDR\_PUBL\_BISTWER**

Address: Operational Base + offset (0x0128)

BIST Word Error Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	DXWER Byte Word Error Indicates the number of word errors on the byte lane. An error on any bit of the data bus including the data mask bit increments the error count.
15:0	RO	0x0000	ACWER Address/Command Word Error Indicates the number of word errors on the address/command lane. An error on any bit of the address/command bus increments the error count.

**DDR\_PUBL\_BISTBER0**

Address: Operational Base + offset (0x012c)

BIST Bit Error Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	ABER Address Bit Error Each group of two bits indicate the bit error count on each of the up to 16 address bits. [1:0] is the error count for A[0], [3:2] for A[1], and so on.

**DDR\_PUBL\_BISTBER1**

Address: Operational Base + offset (0x0130)

BIST Bit Error Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x00	ODTBER ODT Bit Error Each group of two bits indicates the bit error count on each of the up to 4 ODT bits. [1:0] is the error count for ODT[0], [3:2] for ODT[1], and so on.
23:16	RO	0x00	CSBER CS# Bit Error Each group of two bits indicates the bit error count on each of the up to 4 CS# bits. [1:0] is the error count for CS#[0], [3:2] for CS#[1], and so on.
15:8	RO	0x00	CKEBER CKE Bit Error Each group of two bits indicates the bit error count on each of the up to 4 CKE bits. [1:0] is the error count for CKE[0], [3:2] for CKE[1], and so on.
7:6	RO	0x0	WEBER WE# Bit Error Indicates the number of bit errors on WE#
5:0	RO	0x00	BABER Bank Address Bit Error Each group of two bits indicates the bit error count on each of the up to 3 bank address bits. [1:0] is the error count for BA[0], [3:2] for BA[1], and so on.

**DDR\_PUBL\_BISTBER2**

Address: Operational Base + offset (0x0134)

BIST Bit Error Register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	DQBER Data Bit Error The first 16 bits indicate the error count for the first data beat (i.e. the data driven out on DQ[7:0] on the rising edge of DQS). The second 16 bits indicate the error on the second data beat (i.e. the error count of the data driven out on DQ[7:0] on the falling edge of DQS). For each of the 16-bit group, the first 2 bits are for DQ[0], the second for DQ[1], and so on.

**DDR\_PUBL\_BISTWCSR**

Address: Operational Base + offset (0x0138)

BIST Word Count Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	DXWCNT Byte Word Count Indicates the number of words received from the byte lane
15:0	RO	0x0000	ACWCNT Address/Command Word Count Indicates the number of words received from the address/command lane

**DDR\_PUBL\_BISTFWR0**

Address: Operational Base + offset (0x013c)

BIST Fail Word Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28	RO	0x0	ODTWEBS Bit status during a word error for each of the up to 4 ODT bits
27:24	RO	0x0	CSEWS Bit status during a word error for each of the up to 4 CS# bits
23:20	RO	0x0	CKEWEBS Bit status during a word error for each of the up to 4 CKE bits
19	RO	0x0	WEWEBS Bit status during a word error for the WE#
18:16	RO	0x0	BAWEBS Bit status during a word error for each of the up to 3 bank address bits
15:0	RO	0x0000	AWEBS Bit status during a word error for each of the up to 16 address bits

**DDR\_PUBL\_BISTFWR1**

Address: Operational Base + offset (0x0140)

BIST Fail Word Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	TPDWEBS Bit status during a word error for the TPD. LPDDR Only.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
30	RO	0x0	PARWEBS Bit status during a word error for the PAR_IN. Only for DIMM parity support.
29:20	RO	0x0	reserved
19	RO	0x0	CASWEBS Bit status during a word error for the CAS
18	RO	0x0	RASWEBS Bit status during a word error for the RAS
17:16	RO	0x0	DMWEBS Bit status during a word error for the data mask (DM) bit. DMWEBS [0] is for the first DM beat, and DMWEBS [1] is for the second DM beat.
15:0	RO	0x0000	DQWEBS Bit status during a word error for each of the 8 data (DQ) bits. The first 8 bits indicate the status of the first data beat (i.e. the status of the data driven out on DQ[7:0] on the rising edge of DQS). The second 8 bits indicate the status of the second data beat (i.e. the status of the data driven out on DQ[7:0] on the falling edge of DQS). For each of the 8-bit group, the first bit is for DQ[0], the second bit is for DQ[1], and so on.

**DDR\_PUBL\_ZQ0CRO**

Address: Operational Base + offset (0x0180)

ZQ 0 Impedance Control Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	ZQPD ZQ Power Down Powers down, if set, the PZQ cell.
30	RW	0x0	ZCAL Impedance Calibration Trigger A write of '1' to this bit triggers impedance calibration to be performed by the impedance control logic. The impedance calibration trigger bit is self-clearing and returns back to '0' when the calibration is complete.

Bit	Attr	Reset Value	Description
29	RW	0x0	ZCALBYP Impedance Calibration Bypass Disables, if set, impedance calibration of this ZQ control block when impedance calibration is triggered globally using the ZCAL bit of PIR. Impedance calibration of this ZQ block may be triggered manually using ZCAL.
28	RW	0x0	ZDEN Impedance Over-ride Enable When this bit is set, it allows users to directly drive the impedance control using the data programmed in the ZQDATA field. Otherwise, the control is generated automatically by the impedance control logic.
27:0	RW	0x000014a	ZDATA Impedance Over-Ride Data Data used to directly drive the impedance control. ZDATA field mapping for D3R I/Os is as follows: ZDATA[27:20] is reserved and returns zeros on reads ZDATA[19:15] is used to select the pull-up on-die termination impedance ZDATA[14:10] is used to select the pull-down on-die termination impedance ZDATA[9:5] is used to select the pull-up output impedance ZDATA[4:0] is used to select the pull-down output impedance

**DDR\_PUBL\_ZQ0CR1**

Address: Operational Base + offset (0x0184)

ZQ 0 Impedance Control Register 1

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x7b	ZPROG Impedance Divide Ratio Selects the external resistor divide ratio to be used to set the output impedance and the on-die termination as follows: ZPROG[7:4]: On-die termination divide select ZPROG[3:0]: Output impedance divide select

**DDR\_PUBL\_ZQ0SR0**

Address: Operational Base + offset (0x0188)

ZQ 0 Impedance Status Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	ZDONE Impedance Calibration Done Indicates that impedance calibration has completed
30	RO	0x0	ZERR Impedance Calibration Error If set, indicates that there was an error during impedance calibration
29:28	RO	0x0	reserved
27:0	RO	0x00000000	ZCTRL Impedance Control Current value of impedance control. ZCTRL field mapping for D3R I/Os is as follows: ZCTRL[27:20] is reserved and returns zeros on reads ZCTRL[19:15] is used to select the pull-up on-die termination impedance ZCTRL[14:10] is used to select the pull-down on-die termination impedance ZCTRL[9:5] is used to select the pull-up output impedance ZCTRL[4:0] is used to select the pull-down output impedance

**DDR\_PUBL\_ZQ0SR1**

Address: Operational Base + offset (0x018c)

ZQ 0 Impedance Status Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:6	RO	0x0	OPU On-die termination (ODT) pull-up calibration status. Similar status encodings as ZPD.
5:4	RO	0x0	OPD On-die termination (ODT) pull-down calibration status. Similar status encodings as ZPD.
3:2	RO	0x0	ZPU Output impedance pull-up calibration status. Similar status encodings as ZPD.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RO	0x0	ZPD Output impedance pull-down calibration status. Valid status encodings are: 2'b00: Completed with no errors 2'b01: Overflow error 2'b10: Underflow error 2'b11: Calibration in progress

**DDR\_PUBL\_ZQ1CR0**

Address: Operational Base + offset (0x0190)

ZQ 1 Impedance Control Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	ZQPD ZQ Power Down Powers down, if set, the PZQ cell.
30	RW	0x0	ZCAL Impedance Calibration Trigger A write of '1' to this bit triggers impedance calibration to be performed by the impedance control logic. The impedance calibration trigger bit is self-clearing and returns back to '0' when the calibration is complete.
29	RW	0x0	ZCALBYP Impedance Calibration Bypass Disables, if set, impedance calibration of this ZQ control block when impedance calibration is triggered globally using the ZCAL bit of PIR. Impedance calibration of this ZQ block may be triggered manually using ZCAL.
28	RW	0x0	ZDEN Impedance Over-ride Enable When this bit is set, it allows users to directly drive the impedance control using the data programmed in the ZQDATA field. Otherwise, the control is generated automatically by the impedance control logic.

Bit	Attr	Reset Value	Description
27:0	RW	0x000014a	ZDATA Impedance Over-Ride Data Data used to directly drive the impedance control. ZDATA field mapping for D3R I/Os is as follows: ZDATA[27:20] is reserved and returns zeros on reads ZDATA[19:15] is used to select the pull-up on-die termination impedance ZDATA[14:10] is used to select the pull-down on-die termination impedance ZDATA[9:5] is used to select the pull-up output impedance ZDATA[4:0] is used to select the pull-down output impedance

**DDR\_PUBL\_ZQ1CR1**

Address: Operational Base + offset (0x0194)

ZQ 1 Impedance Control Register 1

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x7b	ZPROG Impedance Divide Ratio Selects the external resistor divide ratio to be used to set the output impedance and the on-die termination as follows: ZPROG[7:4]: On-die termination divide select ZPROG[3:0]: Output impedance divide select

**DDR\_PUBL\_ZQ1SR0**

Address: Operational Base + offset (0x0198)

ZQ 1 Impedance Status Register 0

Bit	Attr	Reset Value	Description
31	RO	0x0	ZDONE Impedance Calibration Done Indicates that impedance calibration has completed
30	RO	0x0	ZERR Impedance Calibration Error If set, indicates that there was an error during impedance calibration
29:28	RO	0x0	reserved

Bit	Attr	Reset Value	Description
27:0	RO	0x000000	ZCTRL Impedance Control Current value of impedance control. ZCTRL field mapping for D3R I/Os is as follows: ZCTRL[27:20] is reserved and returns zeros on reads ZCTRL[19:15] is used to select the pull-up on-die termination impedance ZCTRL[14:10] is used to select the pull-down on-die termination impedance ZCTRL[9:5] is used to select the pull-up output impedance ZCTRL[4:0] is used to select the pull-down output impedance

**DDR\_PUBL\_ZQ1SR1**

Address: Operational Base + offset (0x019c)

ZQ 1 Impedance Status Register 1

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:6	RO	0x0	OPU On-die termination (ODT) pull-up calibration status. Similar status encodings as ZPD.
5:4	RO	0x0	OPD On-die termination (ODT) pull-down calibration status. Similar status encodings as ZPD.
3:2	RO	0x0	ZPU Output impedance pull-up calibration status. Similar status encodings as ZPD.
1:0	RO	0x0	ZPD Output impedance pull-down calibration status. Valid status encodings are: 2'b00: Completed with no errors 2'b01: Overflow error 2'b10: Underflow error 2'b11: Calibration in progress

**DDR\_PUBL\_ZQ2CR0**

Address: Operational Base + offset (0x01a0)

ZQ 2 Impedance Control Register 0

Bit	Attr	Reset Value	Description

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	ZQPD ZQ Power Down Powers down, if set, the PZQ cell.
30	RW	0x0	ZCAL Impedance Calibration Trigger A write of '1' to this bit triggers impedance calibration to be performed by the impedance control logic. The impedance calibration trigger bit is self-clearing and returns back to '0' when the calibration is complete.
29	RW	0x0	ZCALBYP Impedance Calibration Bypass Disables, if set, impedance calibration of this ZQ control block when impedance calibration is triggered globally using the ZCAL bit of PIR. Impedance calibration of this ZQ block may be triggered manually using ZCAL.
28	RW	0x0	ZDEN Impedance Over-ride Enable When this bit is set, it allows users to directly drive the impedance control using the data programmed in the ZQDATA field. Otherwise, the control is generated automatically by the impedance control logic.
27:0	RW	0x000014a	ZDATA Impedance Over-Ride Data Data used to directly drive the impedance control. ZDATA field mapping for D3R I/Os is as follows: ZDATA[27:20] is reserved and returns zeros on reads ZDATA[19:15] is used to select the pull-up on-die termination impedance ZDATA[14:10] is used to select the pull-down on-die termination impedance ZDATA[9:5] is used to select the pull-up output impedance ZDATA[4:0] is used to select the pull-down output impedance

**DDR\_PUBL\_ZQ2CR1**

Address: Operational Base + offset (0x01a4)

ZQ 2 Impedance Control Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x7b	ZPROG Impedance Divide Ratio Selects the external resistor divide ratio to be used to set the output impedance and the on-die termination as follows: ZPROG[7:4]: On-die termination divide select ZPROG[3:0]: Output impedance divide select

**DDR\_PUBL\_ZQ2SR0**

Address: Operational Base + offset (0x01a8)

ZQ 2 Impedance Status Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	ZDONE Impedance Calibration Done Indicates that impedance calibration has completed
30	RO	0x0	ZERR Impedance Calibration Error If set, indicates that there was an error during impedance calibration
29:28	RO	0x0	reserved
27:0	RO	0x00000000	ZCTRL Impedance Control Current value of impedance control. ZCTRL field mapping for D3R I/Os is as follows: ZCTRL[27:20] is reserved and returns zeros on reads ZCTRL[19:15] is used to select the pull-up on-die termination impedance ZCTRL[14:10] is used to select the pull-down on-die termination impedance ZCTRL[9:5] is used to select the pull-up output impedance ZCTRL[4:0] is used to select the pull-down output impedance

**DDR\_PUBL\_ZQ2SR1**

Address: Operational Base + offset (0x01ac)

ZQ 2 Impedance Status Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved

Bit	Attr	Reset Value	Description
7:6	RO	0x0	OPU On-die termination (ODT) pull-up calibration status. Similar status encodings as ZPD.
5:4	RO	0x0	OPD On-die termination (ODT) pull-down calibration status. Similar status encodings as ZPD.
3:2	RO	0x0	ZPU Output impedance pull-up calibration status. Similar status encodings as ZPD.
1:0	RO	0x0	ZPD Output impedance pull-down calibration status. Valid status encodings are: 2'b00: Completed with no errors 2'b01: Overflow error 2'b10: Underflow error 2'b11: Calibration in progress

**DDR\_PUBL\_ZQ3CRO**

Address: Operational Base + offset (0x01b0)

ZQ 3 Impedance Control Register 0

Bit	Attr	Reset Value	Description
31	RW	0x0	ZQPD ZQ Power Down Powers down, if set, the PZQ cell.
30	RW	0x0	ZCAL Impedance Calibration Trigger A write of '1' to this bit triggers impedance calibration to be performed by the impedance control logic. The impedance calibration trigger bit is self-clearing and returns back to '0' when the calibration is complete.
29	RW	0x0	ZCALBYP Impedance Calibration Bypass Disables, if set, impedance calibration of this ZQ control block when impedance calibration is triggered globally using the ZCAL bit of PIR. Impedance calibration of this ZQ block may be triggered manually using ZCAL.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
28	RW	0x0	ZDEN Impedance Over-ride Enable When this bit is set, it allows users to directly drive the impedance control using the data programmed in the ZQDATA field. Otherwise, the control is generated automatically by the impedance control logic
27:0	RW	0x000014a	ZDATA Impedance Over-Ride Data Data used to directly drive the impedance control. ZDATA field mapping for D3R I/Os is as follows: ZDATA[27:20] is reserved and returns zeros on reads ZDATA[19:15] is used to select the pull-up on-die termination impedance ZDATA[14:10] is used to select the pull-down on-die termination impedance ZDATA[9:5] is used to select the pull-up output impedance ZDATA[4:0] is used to select the pull-down output impedance

**DDR\_PUBL\_ZQ3CR1**

Address: Operational Base + offset (0x01b4)

ZQ 3 Impedance Control Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x7b	ZPROG Impedance Divide Ratio Selects the external resistor divide ratio to be used to set the output impedance and the on-die termination as follows: ZPROG[7:4]: On-die termination divide select ZPROG[3:0]: Output impedance divide select

**DDR\_PUBL\_ZQ3SR0**

Address: Operational Base + offset (0x01b8)

ZQ 3 Impedance Status Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	ZDONE Impedance Calibration Done Indicates that impedance calibration has completed

Bit	Attr	Reset Value	Description
30	RO	0x0	ZERR Impedance Calibration Error If set, indicates that there was an error during impedance calibration
29:28	RO	0x0	reserved
27:0	RO	0x00000000	ZCTRL Impedance Control Current value of impedance control. ZCTRL field mapping for D3R I/Os is as follows: ZCTRL[27:20] is reserved and returns zeros on reads ZCTRL[19:15] is used to select the pull-up on-die termination impedance ZCTRL[14:10] is used to select the pull-down on-die termination impedance ZCTRL[9:5] is used to select the pull-up output impedance ZCTRL[4:0] is used to select the pull-down output impedance

**DDR\_PUBL\_ZQ3SR1**

Address: Operational Base + offset (0x01bc)

ZQ 3 Impedance Status Register 1

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:6	RO	0x0	OPU On-die termination (ODT) pull-up calibration status. Similar status encodings as ZPD.
5:4	RO	0x0	OPD On-die termination (ODT) pull-down calibration status. Similar status encodings as ZPD.
3:2	RO	0x0	ZPU Output impedance pull-up calibration status. Similar status encodings as ZPD.
1:0	RO	0x0	ZPD Output impedance pull-down calibration status. Valid status encodings are: 2'b00: Completed with no errors 2'b01: Overflow error 2'b10: Underflow error 2'b11: Calibration in progress

**DDR\_PUBL\_DX0GCR**

Address: Operational Base + offset (0x01c0)

DATX8 0 General Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
16:14 19:17	RW	0x3 0x3	<p>R0RVSL R1RVSL Rank n ITMD Read Valid System Latency Used to specify the read valid system latency relative to the ideal placement of the ITMD read valid signal when DXCCR.RVSEL is set to 0. Power-up default is 011 (i.e. ideal placement of the read valid signal). The RVSL fields are initially set by the PUB during automatic read valid training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each rank. R0RVSL controls the latency of rank 0, R1RVSL controls rank 1. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: read valid system latency = ideal placement - 3</li> <li>3'b001: read valid system latency = ideal placement - 2</li> <li>3'b010: read valid system latency = ideal placement - 1</li> <li>3'b011: read valid system latency = ideal placement</li> <li>3'b100: read valid system latency = ideal placement + 1</li> <li>3'b101: read valid system latency = ideal placement + 2</li> <li>3'b110: read valid system latency = ideal placement + 3</li> <li>3'b111: Reserved</li> </ul>
13	RW	0x0	<p>RTTOAL RTT On Additive Latency Indicates when the ODT control of DQ/DQS SSTL I/Os is set to the value in DQODT/DQSODT during read cycles. Valid values are:</p> <ul style="list-style-type: none"> <li>1'b0: ODT control is set to DQSODT/DQODT almost two cycles before read data preamble</li> <li>1'b1: ODT control is set to DQSODT/DQODT almost one cycle before read data preamble</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12:11	RW	0x0	RTTOH RTT Output Hold Indicates the number of clock cycles (from 0 to 3) after the read data postamble for which ODT control should remain set to DQSODT for DQS or DQODT for DQ/DM before disabling it (setting it to '0' when using dynamic ODT control. ODT is disabled almost RTTOH clock cycles after the read postamble.
10	RW	0x1	DQRTT DQ Dynamic RTT Control Indicates, if set, that the ODT control of DQ/DM SSTL I/Os be dynamically controlled by setting it to the value in DQODT during reads and disabling it (setting it to '0' during any other cycle. If this bit is not set, then the ODT control of DQ SSTL I/Os is always set to the value in DQODT.
9	RW	0x1	DQSRTT DQS Dynamic RTT Control Indicates, if set, that the ODT control of DQS SSTL I/Os be dynamically controlled by setting it to the value in DQSODT during reads and disabling it (setting it to '0' during any other cycle. If this bit is not set, then the ODT control of DQS SSTL I/Os is always set to the value in DQSODT field.
8:7	RW	0x1	DSEN Write DQS Enable Controls whether the write DQS going to the SDRAM is enabled (toggling) or disabled (static value) and whether the DQS is inverted. DQS# is always the inversion of DQS. These values are valid only when DQS/DQS# output enable is on, otherwise the DQS/DQS# is tristated. Valid settings are: 2'b00: DQS disabled (Driven to constant 0) 2'b01: DQS toggling with inverted polarity 2'b10: DQS toggling with normal polarity (This should be the default setting) 2'b11: DQS disabled (Driven to constant 1)
6	RW	0x0	DQSRPD DQSR Power Down Powers down, if set, the PDQSR cell. This bit is ORed with the common PDR configuration bit.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	DXPDR Data Power Down Receiver Powers down, when set, the input receiver on I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the common PDR configuration bit.
4	RW	0x0	DXPDD Data Power Down Driver Powers down, when set, the output driver on I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the common PDD configuration bit.
3	RW	0x0	DXIOM Data I/O Mode Selects SSTL mode (when set to 0) or CMOS mode (when set to 1) of the I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the IOM configuration bit of the individual DATX8.
2	RW	0x0	DQODT Data On-Die Termination Enables, when set, the on-die termination on the I/O for DQ and DM pins of the byte. This bit is ORed with the common DATX8 ODT configuration bit.
1	RW	0x0	DQSODT DQS On-Die Termination Enables, when set, the on-die termination on the I/O for DQS/DQS# pin of the byte. This bit is ORed with the common DATX8 ODT configuration bit.
0	RW	0x1	DXEN Data Byte Enable Enables if set the DATX8 and SSTL I/Os used on the data byte. Setting this bit to '0' disables the byte, i.e. the byte SSTL I/Os are put in power-down mode and the DLL in the DATX8 is put in bypass mode.

**DDR\_PUBL\_DX0GSR0**

Address: Operational Base + offset (0x01c4)

DATX8 0 General Status Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:13	RO	0x000	DTPASS DQS Gate Training Pass Count The number of passing configurations during DQS gate training. Bits [2:0] are for rank 0, bits [5:3] for rank 1, and so on.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RO	0x0	reserved
11:8	RO	0x0	DTIERR DQS Gate Training Intermittent Error If set, indicates that there was an intermittent error during DQS gate training of the byte, such as a pass was followed by a fail then followed by another pass. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
7:4	RO	0x0	DTERR DQS Gate Training Error If set, indicates that a valid DQS gating window could not be found during DQS gate training of the byte. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
3:0	RO	0x0	DTDONE Data Training Done Indicates, if set, that the byte has finished doing data training. Bit [0] is for rank 0, bit 1 for rank 1, and so on.

**DDR\_PUBL\_DX0GSR1**

Address: Operational Base + offset (0x01c8)

DATX8 0 General Status Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	RVPASS Read Valid Training Pass Count The number of passing configurations during read valid training. Bits [2:0] are for rank 0, bits [5:3] for rank 1, and so on.
19:16	RO	0x0	RVIERR Read Valid Training Intermittent Error If set, indicates that there was an intermittent error during read valid training of the byte, such as a pass was followed by a fail then followed by another pass. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
15:12	RO	0x0	RVERR Read Valid Training Error If set, indicates that a valid read valid placement could not be found during read valid training of the byte. Bit [0] is for rank 0, bit 1 for rank 1, and so on.

Bit	Attr	Reset Value	Description
11:4	RO	0x00	DQSDFT DQS Drift Used to report the drift on the read data strobe of the data byte. Valid settings are: 2'b00: No drift 2'b01: 90 deg drift 2'b10: 180 deg drift 2'b11: 270 deg drift or more Bits [1:0] are for rank 0, bits [3:2] for rank 1, and so on.
3:0	RO	0x0	DFTERR DQS Drift Error If set, indicates that the byte read data strobe has drifted by more than or equal to the drift limit set in the PHY General Configuration Register (PGCR). Bit [0] is for rank 0, bit 1 for rank 1, and so on.

**DDR\_PUBL\_DX0DLLCR**

Address: Operational Base + offset (0x01cc)

DATX8 0 DLL Control Register

Bit	Attr	Reset Value	Description
31	RW	0x0	DLLDIS DLL Disable A disabled DLL is bypassed. Default ('0') is DLL enabled
30	RW	0x1	DLLSRST DLL Soft Rest Soft resets the byte DLL by driving the DLL soft reset pin
29:20	RO	0x0	reserved
19	RW	0x0	SDLBMODE Slave DLL Loopback Mode If this bit is set, the slave DLL is put in loopback mode in which there is no 90 degrees phase shift on read DQS/DQS#. This bit must be set when operating the byte PHYs in loopback mode such as during BIST loopback. Applicable only to PHYs that have this feature. Refer to PHY databook.
18	RW	0x0	ATESTEN Analog Test Enable Enables the analog test signal to be output on the DLL analog test output (test_out_a). The DLL analog test output is tri-stated when this bit is '0'.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17:14	RW	0x0	<p>SDPHASE Slave DLL Phase Trim Selects the phase difference between the input clock and the corresponding output clock of the slave DLL. Valid settings:</p> <p>4'b0000: 90 4'b0001: 72 4'b0010: 54 4'b0011: 36 4'b0100: 108 4'b0101: 90 4'b0110: 72 4'b0111: 54 4'b1000: 126 4'b1001: 108 4'b1010: 90 4'b1011: 72 4'b1100: 144 4'b1101: 126 4'b1110: 108 4'b1111: 90</p>
13:12	RW	0x0	<p>SSTART Slave Auto Start-Up Used to control how the slave DLL starts up relative to the master DLL locking:</p> <p>2'b0X: Slave DLL automatically starts up once the master DLL has achieved lock 2'b10: The automatic startup of the slave DLL is disabled; the phase detector is disabled 2'b11: The automatic startup of the slave DLL is disabled; the phase detector is enabled</p>
11:9	RW	0x0	<p>MFWDLY Master Feed-Forward Delay Trim Used to trim the delay in the master DLL feed-forward path:</p> <p>3'b000: minimum delay 3'b111: maximum delay</p>
8:6	RW	0x0	<p>MFBFDLY Master Feed-Back Delay Trim Used to trim the delay in the master DLL feedback path:</p> <p>3'b000: minimum delay 3'b111: maximum delay</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:3	RW	0x0	SFWDLY Slave Feed-Forward Delay Trim Used to trim the delay in the slave DLL feed-forward path: 3'b000: minimum delay 3'b111: maximum delay
2:0	RW	0x0	SFBDLY Slave Feed-Back Delay Trim Used to trim the delay in the slave DLL feedback path: 3'b000: minimum delay 3'b111: maximum delay

**DDR\_PUBL\_DX0DQTR**

Address: Operational Base + offset (0x01d0)

DATX8 0 DQ Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RW	0xf	DQDLY7 DQ7 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte. The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are: 2'b00: nominal delay 2'b01: nominal delay + 1 step 2'b10: nominal delay + 2 steps 2'b11: nominal delay + 3 steps

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27:24	RW	0xf	<p>DQDLY6 DQ6 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
23:20	RW	0xf	<p>DQDLY5 DQ5 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19:16	RW	0xf	<p>DQDLY4 DQ4 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
15:12	RW	0xf	<p>DQDLY3 DQ3 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:8	RW	0xf	<p>DQDLY2 DQ2 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
7:4	RW	0xf	<p>DQDLY1 DQ1 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

Bit	Attr	Reset Value	Description
3:0	RW	0xf	<p>DQDLY0 DQ0 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

**DDR\_PUBL\_DX0DQSTR**

Address: Operational Base + offset (0x01d4)

DATX8 0 DQS Timing Register

Bit	Attr	Reset Value	Description
31:30	RO	0x0	reserved
29:26	RW	0xf	<p>DMDLY DM Delay Used to adjust the delay of the data mask relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. The lower two bits of the DQMDLY controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b.</p> <p>Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

Bit	Attr	Reset Value	Description
25:23	RW	0x3	<p>DQSNDLY DQS# Delay Used to adjust the delay of the data strobes relative to the nominal delay that is matched to the delay of the data bit through the slave DLL and clock tree. DQSDLY control the delay on DQS strobe and DQSNDLY control the delay on DQS#. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: nominal delay - 3 steps</li> <li>3'b001: nominal delay - 2 steps</li> <li>3'b010: nominal delay - 1 step</li> <li>3'b011: nominal delay</li> <li>3'b100: nominal delay + 1 step</li> <li>3'b101: nominal delay + 2 steps</li> <li>3'b110: nominal delay + 3 steps</li> <li>3'b111: nominal delay + 4 steps</li> </ul>
22:20	RW	0x3	<p>DQSDLY DQS Delay Used to adjust the delay of the data strobes relative to the nominal delay that is matched to the delay of the data bit through the slave DLL and clock tree. DQSDLY control the delay on DQS strobe and DQSNDLY control the delay on DQS#. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: nominal delay - 3 steps</li> <li>3'b001: nominal delay - 2 steps</li> <li>3'b010: nominal delay - 1 step</li> <li>3'b011: nominal delay</li> <li>3'b100: nominal delay + 1 step</li> <li>3'b101: nominal delay + 2 steps</li> <li>3'b110: nominal delay + 3 steps</li> <li>3'b111: nominal delay + 4 steps</li> </ul>
19:16	RO	0x0	reserved

Bit	Attr	Reset Value	Description
15:14	RW	0x2	<p>R1DGPS</p> <p>Rank 1 DQS Gating Phase Select</p> <p>Selects the clock used to enable the data strobes during read so that the value of the data strobes before and after the preamble/postamble are filtered out. The RnDGPS fields are initially set by the PUBL during automatic DQS data training and subsequently updated during data strobe drift compensation. However, these values can be overwritten by a direct write to this register, and the automatic update during DQS drift compensation can be disabled using the PHY General Configuration Register (PGCR). Every two bits of this register control the DQS gating for each of the (up to) four ranks. Valid values for each 2-bit RnDGPS field are:</p> <ul style="list-style-type: none"> <li>2'b00: 90 deg clock (clk90)</li> <li>2'b01: 180 deg clock (clk180)</li> <li>2'b10: 270 deg clock (clk270)</li> <li>2'b11: 360 deg clock (clk0)</li> </ul>
13:12	RW	0x2	<p>R0DGPS</p> <p>Rank 0 DQS Gating Phase Select</p> <p>Selects the clock used to enable the data strobes during read so that the value of the data strobes before and after the preamble/postamble are filtered out. The RnDGPS fields are initially set by the PUBL during automatic DQS data training and subsequently updated during data strobe drift compensation. However, these values can be overwritten by a direct write to this register, and the automatic update during DQS drift compensation can be disabled using the PHY General Configuration Register (PGCR). Every two bits of this register control the DQS gating for each of the (up to) four ranks. Valid values for each 2-bit RnDGPS field are:</p> <ul style="list-style-type: none"> <li>2'b00: 90 deg clock (clk90)</li> <li>2'b01: 180 deg clock (clk180)</li> <li>2'b10: 270 deg clock (clk270)</li> <li>2'b11: 360 deg clock (clk0)</li> </ul>
11:6	RO	0x0	reserved

Bit	Attr	Reset Value	Description
5:3	RW	0x0	<p>R1DGSL Rank 1 DQS Gating System Latency Used to increase the number of clock cycles needed to expect valid DDR read data by up to five extra clock cycles. This is used to compensate for board delays and other system delays. Power-up default is 000 (i.e. no extra clock cycles required). The SL fields are initially set by the PUBL during automatic DQS data training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each of the (up to) four ranks. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: No extra clock cycles</li> <li>3'b001: 1 extra clock cycle</li> <li>3'b010: 2 extra clock cycles</li> <li>3'b011: 3 extra clock cycles</li> <li>3'b100: 4 extra clock cycles</li> <li>3'b101: 5 extra clock cycles</li> <li>3'b110: Reserved</li> <li>3'b111: Reserved</li> </ul>
2:0	RW	0x0	<p>R0DGSL Rank 0 DQS Gating System Latency Used to increase the number of clock cycles needed to expect valid DDR read data by up to five extra clock cycles. This is used to compensate for board delays and other system delays. Power-up default is 000 (i.e. no extra clock cycles required). The SL fields are initially set by the PUBL during automatic DQS data training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each of the (up to) four ranks. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: No extra clock cycles</li> <li>3'b001: 1 extra clock cycle</li> <li>3'b010: 2 extra clock cycles</li> <li>3'b011: 3 extra clock cycles</li> <li>3'b100: 4 extra clock cycles</li> <li>3'b101: 5 extra clock cycles</li> <li>3'b110: Reserved</li> <li>3'b111: Reserved</li> </ul>

**DDR\_PUBL\_DX1GCR**

Address: Operational Base + offset (0x0200)  
DATX8 1 General Configuration Register

Bit	Attr	Reset Value	Description
31:20	RO	0x0	reserved
16:14 19:17	RW	0x3 0x3	<p>R0RVSL R1RVSL</p> <p>Rank n ITMD Read Valid System Latency Used to specify the read valid system latency relative to the ideal placement of the ITMD read valid signal when DXCCR.RVSEL is set to 0.</p> <p>Power-up default is 011 (i.e. ideal placement of the read valid signal). The RVSL fields are initially set by the PUB during automatic read valid training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each rank. R0RVSL controls the latency of rank 0, R1RVSL controls rank 1. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: read valid system latency = ideal placement - 3</li> <li>3'b001: read valid system latency = ideal placement - 2</li> <li>3'b010: read valid system latency = ideal placement - 1</li> <li>3'b011: read valid system latency = ideal placement</li> <li>3'b100: read valid system latency = ideal placement + 1</li> <li>3'b101: read valid system latency = ideal placement + 2</li> <li>3'b110: read valid system latency = ideal placement + 3</li> <li>3'b111: Reserved</li> </ul>
13	RW	0x0	<p>RTTOAL</p> <p>RTT On Additive Latency</p> <p>Indicates when the ODT control of DQ/DQS SSTL I/Os is set to the value in DQODT/DQSODT during read cycles. Valid values are:</p> <ul style="list-style-type: none"> <li>1'b0: ODT control is set to DQSODT/DQODT almost two cycles before read data preamble</li> <li>1'b1: ODT control is set to DQSODT/DQODT almost one cycle before read data preamble</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12:11	RW	0x0	RTTOH RTT Output Hold Indicates the number of clock cycles (from 0 to 3) after the read data postamble for which ODT control should remain set to DQSODT for DQS or DQODT for DQ/DM before disabling it (setting it to '0' when using dynamic ODT control. ODT is disabled almost RTTOH clock cycles after the read postamble.
10	RW	0x1	DQRTT DQ Dynamic RTT Control Indicates, if set, that the ODT control of DQ/DM SSTL I/Os be dynamically controlled by setting it to the value in DQODT during reads and disabling it (setting it to '0' during any other cycle. If this bit is not set, then the ODT control of DQ SSTL I/Os is always set to the value in DQODT.
9	RW	0x1	DQSRTT DQS Dynamic RTT Control Indicates, if set, that the ODT control of DQS SSTL I/Os be dynamically controlled by setting it to the value in DQSODT during reads and disabling it (setting it to '0' during any other cycle. If this bit is not set, then the ODT control of DQS SSTL I/Os is always set to the value in DQSODT field.
8:7	RW	0x1	DSEN Write DQS Enable Controls whether the write DQS going to the SDRAM is enabled (toggling) or disabled (static value) and whether the DQS is inverted. DQS# is always the inversion of DQS. These values are valid only when DQS/DQS# output enable is on, otherwise the DQS/DQS# is tristated. Valid settings are: 2'b00: DQS disabled (Driven to constant 0) 2'b01: DQS toggling with inverted polarity 2'b10: DQS toggling with normal polarity (This should be the default setting) 2'b11: DQS disabled (Driven to constant 1)
6	RW	0x0	DQSRPD DQSR Power Down Powers down, if set, the PDQSR cell. This bit is ORed with the common PDR configuration bit.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	DXPDR Data Power Down Receiver Powers down, when set, the input receiver on I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the common PDR configuration bit.
4	RW	0x0	DXPDD Data Power Down Driver Powers down, when set, the output driver on I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the common PDD configuration bit.
3	RW	0x0	DXIOM Data I/O Mode Selects SSTL mode (when set to 0) or CMOS mode (when set to 1) of the I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the IOM configuration bit of the individual DATX8.
2	RW	0x0	DQODT Data On-Die Termination Enables, when set, the on-die termination on the I/O for DQ and DM pins of the byte. This bit is ORed with the common DATX8 ODT configuration bit.
1	RW	0x0	DQSODT DQS On-Die Termination Enables, when set, the on-die termination on the I/O for DQS/DQS# pin of the byte. This bit is ORed with the common DATX8 ODT configuration bit.
0	RW	0x1	DXEN Data Byte Enable Enables if set the DATX8 and SSTL I/Os used on the data byte. Setting this bit to '0' disables the byte, i.e. the byte SSTL I/Os are put in power-down mode and the DLL in the DATX8 is put in bypass mode.

**DDR\_PUBL\_DX1GSR0**

Address: Operational Base + offset (0x0204)

DATX8 1 General Status Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:13	RO	0x000	DTPASS DQS Gate Training Pass Count The number of passing configurations during DQS gate training. Bits [2:0] are for rank 0, bits [5:3] for rank 1, and so on.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RO	0x0	reserved
11:8	RO	0x0	DTIERR DQS Gate Training Intermittent Error If set, indicates that there was an intermittent error during DQS gate training of the byte, such as a pass was followed by a fail then followed by another pass. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
7:4	RO	0x0	DTERR DQS Gate Training Error If set, indicates that a valid DQS gating window could not be found during DQS gate training of the byte. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
3:0	RO	0x0	DTDONE Data Training Done Indicates, if set, that the byte has finished doing data training. Bit [0] is for rank 0, bit 1 for rank 1, and so on.

**DDR\_PUBL\_DX1GSR1**

Address: Operational Base + offset (0x0208)

DATX8 1 General Status Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	RVPASS Read Valid Training Pass Count The number of passing configurations during read valid training. Bits [2:0] are for rank 0, bits [5:3] for rank 1, and so on.
19:16	RO	0x0	RVIERR Read Valid Training Intermittent Error If set, indicates that there was an intermittent error during read valid training of the byte, such as a pass was followed by a fail then followed by another pass. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
15:12	RO	0x0	RVERR Read Valid Training Error If set, indicates that a valid read valid placement could not be found during read valid training of the byte. Bit [0] is for rank 0, bit 1 for rank 1, and so on.

Bit	Attr	Reset Value	Description
11:4	RO	0x00	DQSDFT DQS Drift Used to report the drift on the read data strobe of the data byte. Valid settings are: 2'b00: No drift 2'b01: 90 deg drift 2'b10: 180 deg drift 2'b11: 270 deg drift or more Bits [1:0] are for rank 0, bits [3:2] for rank 1, and so on.
3:0	RO	0x0	DFTERR DQS Drift Error If set, indicates that the byte read data strobe has drifted by more than or equal to the drift limit set in the PHY General Configuration Register (PGCR). Bit [0] is for rank 0, bit 1 for rank 1, and so on.

**DDR\_PUBL\_DX1DLLCR**

Address: Operational Base + offset (0x020c)

DATX8 1 DLL Control Register

Bit	Attr	Reset Value	Description
31	RW	0x0	DLLDIS DLL Disable A disabled DLL is bypassed. Default ('0') is DLL enabled
30	RW	0x1	DLLSRST DLL Soft Rest Soft resets the byte DLL by driving the DLL soft reset pin
29:20	RO	0x0	reserved
19	RW	0x0	SDLBMODE Slave DLL Loopback Mode If this bit is set, the slave DLL is put in loopback mode in which there is no 90 degrees phase shift on read DQS/DQS#. This bit must be set when operating the byte PHYs in loopback mode such as during BIST loopback. Applicable only to PHYs that have this feature. Refer to PHY databook.
18	RW	0x0	ATESTEN Analog Test Enable Enables the analog test signal to be output on the DLL analog test output (test_out_a). The DLL analog test output is tri-stated when this bit is '0'.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17:14	RW	0x0	<p>SDPHASE Slave DLL Phase Trim Selects the phase difference between the input clock and the corresponding output clock of the slave DLL. Valid settings:</p> <p>4'b0000: 90 4'b0001: 72 4'b0010: 54 4'b0011: 36 4'b0100: 108 4'b0101: 90 4'b0110: 72 4'b0111: 54 4'b1000: 126 4'b1001: 108 4'b1010: 90 4'b1011: 72 4'b1100: 144 4'b1101: 126 4'b1110: 108 4'b1111: 90</p>
13:12	RW	0x0	<p>SSTART Slave Auto Start-Up Used to control how the slave DLL starts up relative to the master DLL locking:</p> <p>2'b0X: Slave DLL automatically starts up once the master DLL has achieved lock 2'b10: The automatic startup of the slave DLL is disabled; the phase detector is disabled 2'b11: The automatic startup of the slave DLL is disabled; the phase detector is enabled</p>
11:9	RW	0x0	<p>MFWDLY Master Feed-Forward Delay Trim Used to trim the delay in the master DLL feed-forward path:</p> <p>3'b000: minimum delay 3'b111: maximum delay</p>
8:6	RW	0x0	<p>MFBFDLY Master Feed-Back Delay Trim Used to trim the delay in the master DLL feedback path:</p> <p>3'b000: minimum delay 3'b111: maximum delay</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:3	RW	0x0	SFWDLY Slave Feed-Forward Delay Trim Used to trim the delay in the slave DLL feed-forward path: 3'b000: minimum delay 3'b111: maximum delay
2:0	RW	0x0	SFBDLY Slave Feed-Back Delay Trim Used to trim the delay in the slave DLL feedback path: 3'b000: minimum delay 3'b111: maximum delay

**DDR\_PUBL\_DX1DQTR**

Address: Operational Base + offset (0x0210)

DATX8 1 DQ Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RW	0xf	DQDLY7 DQ7 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte. The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are: 2'b00: nominal delay 2'b01: nominal delay + 1 step 2'b10: nominal delay + 2 steps 2'b11: nominal delay + 3 steps

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27:24	RW	0xf	<p>DQDLY6 DQ6 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
23:20	RW	0xf	<p>DQDLY5 DQ5 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19:16	RW	0xf	<p>DQDLY4 DQ4 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
15:12	RW	0xf	<p>DQDLY3 DQ3 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:8	RW	0xf	<p>DQDLY2 DQ2 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
7:4	RW	0xf	<p>DQDLY1 DQ1 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

Bit	Attr	Reset Value	Description
3:0	RW	0xf	<p>DQDLY0 DQ0 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

**DDR\_PUBL\_DX1DQSTR**

Address: Operational Base + offset (0x0214)

DATX8 1 DQS Timing Register

Bit	Attr	Reset Value	Description
31:30	RO	0x0	reserved
29:26	RW	0xf	<p>DMDLY Used to adjust the delay of the data mask relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. The lower two bits of the DQMDLY controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b.</p> <p>Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
25:23	RW	0x3	<p>DQSNDLY DQS# Delay Used to adjust the delay of the data strobes relative to the nominal delay that is matched to the delay of the data bit through the slave DLL and clock tree. DQSDLY control the delay on DQS strobe and DQSNDLY control the delay on DQS#. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: nominal delay - 3 steps</li> <li>3'b001: nominal delay - 2 steps</li> <li>3'b010: nominal delay - 1 step</li> <li>3'b011: nominal delay</li> <li>3'b100: nominal delay + 1 step</li> <li>3'b101: nominal delay + 2 steps</li> <li>3'b110: nominal delay + 3 steps</li> <li>3'b111: nominal delay + 4 steps</li> </ul>
22:20	RW	0x3	<p>DQSDLY DQS Delay Used to adjust the delay of the data strobes relative to the nominal delay that is matched to the delay of the data bit through the slave DLL and clock tree. DQSDLY control the delay on DQS strobe and DQSNDLY control the delay on DQS#. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: nominal delay - 3 steps</li> <li>3'b001: nominal delay - 2 steps</li> <li>3'b010: nominal delay - 1 step</li> <li>3'b011: nominal delay</li> <li>3'b100: nominal delay + 1 step</li> <li>3'b101: nominal delay + 2 steps</li> <li>3'b110: nominal delay + 3 steps</li> <li>3'b111: nominal delay + 4 steps</li> </ul>
19:16	RO	0x0	reserved

Bit	Attr	Reset Value	Description
15:14	RW	0x2	<p>R1DGPS</p> <p>Rank 1 DQS Gating Phase Select</p> <p>Selects the clock used to enable the data strobes during read so that the value of the data strobes before and after the preamble/postamble are filtered out. The RnDGPS fields are initially set by the PUBL during automatic DQS data training and subsequently updated during data strobe drift compensation. However, these values can be overwritten by a direct write to this register, and the automatic update during DQS drift compensation can be disabled using the PHY General Configuration Register (PGCR). Every two bits of this register control the DQS gating for each of the (up to) four ranks. Valid values for each 2-bit RnDGPS field are:</p> <ul style="list-style-type: none"> <li>2'b00: 90 deg clock (clk90)</li> <li>2'b01: 180 deg clock (clk180)</li> <li>2'b10: 270 deg clock (clk270)</li> <li>2'b11: 360 deg clock (clk0)</li> </ul>
13:12	RW	0x2	<p>R0DGPS</p> <p>Rank 0 DQS Gating Phase Select</p> <p>Selects the clock used to enable the data strobes during read so that the value of the data strobes before and after the preamble/postamble are filtered out. The RnDGPS fields are initially set by the PUBL during automatic DQS data training and subsequently updated during data strobe drift compensation. However, these values can be overwritten by a direct write to this register, and the automatic update during DQS drift compensation can be disabled using the PHY General Configuration Register (PGCR). Every two bits of this register control the DQS gating for each of the (up to) four ranks. Valid values for each 2-bit RnDGPS field are:</p> <ul style="list-style-type: none"> <li>2'b00: 90 deg clock (clk90)</li> <li>2'b01: 180 deg clock (clk180)</li> <li>2'b10: 270 deg clock (clk270)</li> <li>2'b11: 360 deg clock (clk0)</li> </ul>
11:6	RO	0x0	reserved

Bit	Attr	Reset Value	Description
5:3	RW	0x0	<p>R1DGSL</p> <p>Rank 1 DQS Gating System Latency</p> <p>Used to increase the number of clock cycles needed to expect valid DDR read data by up to five extra clock cycles. This is used to compensate for board delays and other system delays. Power-up default is 000 (i.e. no extra clock cycles required). The SL fields are initially set by the PUBL during automatic DQS data training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each of the (up to) four ranks. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: No extra clock cycles</li> <li>3'b001: 1 extra clock cycle</li> <li>3'b010: 2 extra clock cycles</li> <li>3'b011: 3 extra clock cycles</li> <li>3'b100: 4 extra clock cycles</li> <li>3'b101: 5 extra clock cycles</li> <li>3'b110: Reserved</li> <li>3'b111: Reserved</li> </ul>
2:0	RW	0x0	<p>R0DGSL</p> <p>Rank 0 DQS Gating System Latency</p> <p>Used to increase the number of clock cycles needed to expect valid DDR read data by up to five extra clock cycles. This is used to compensate for board delays and other system delays. Power-up default is 000 (i.e. no extra clock cycles required). The SL fields are initially set by the PUBL during automatic DQS data training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each of the (up to) four ranks. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: No extra clock cycles</li> <li>3'b001: 1 extra clock cycle</li> <li>3'b010: 2 extra clock cycles</li> <li>3'b011: 3 extra clock cycles</li> <li>3'b100: 4 extra clock cycles</li> <li>3'b101: 5 extra clock cycles</li> <li>3'b110: Reserved</li> <li>3'b111: Reserved</li> </ul>

**DDR\_PUBL\_DX2GCR**

Address: Operational Base + offset (0x0240)  
 DATX8 2 General Configuration Register

Bit	Attr	Reset Value	Description
31:20	RO	0x0	reserved
16:14 19:17	RW	0x3 0x3	<p>R0RVSL R1RVSL</p> <p>Rank n ITMD Read Valid System Latency Used to specify the read valid system latency relative to the ideal placement of the ITMD read valid signal when DXCCR.RVSEL is set to 0.</p> <p>Power-up default is 011 (i.e. ideal placement of the read valid signal). The RVSL fields are initially set by the PUB during automatic read valid training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each rank. R0RVSL controls the latency of rank 0, R1RVSL controls rank 1. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: read valid system latency = ideal placement - 3</li> <li>3'b001: read valid system latency = ideal placement - 2</li> <li>3'b010: read valid system latency = ideal placement - 1</li> <li>3'b011: read valid system latency = ideal placement</li> <li>3'b100: read valid system latency = ideal placement + 1</li> <li>3'b101: read valid system latency = ideal placement + 2</li> <li>3'b110: read valid system latency = ideal placement + 3</li> <li>3'b111: Reserved</li> </ul>
13	RW	0x0	<p>RTTOAL</p> <p>RTT On Additive Latency</p> <p>Indicates when the ODT control of DQ/DQS SSTL I/Os is set to the value in DQODT/DQSODT during read cycles. Valid values are:</p> <ul style="list-style-type: none"> <li>1'b0: ODT control is set to DQSODT/DQODT almost two cycles before read data preamble</li> <li>1'b1: ODT control is set to DQSODT/DQODT almost one cycle before read data preamble</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12:11	RW	0x0	RTTOH RTT Output Hold Indicates the number of clock cycles (from 0 to 3) after the read data postamble for which ODT control should remain set to DQSODT for DQS or DQODT for DQ/DM before disabling it (setting it to '0' when using dynamic ODT control. ODT is disabled almost RTTOH clock cycles after the read postamble.
10	RW	0x1	DQRTT DQ Dynamic RTT Control Indicates, if set, that the ODT control of DQ/DM SSTL I/Os be dynamically controlled by setting it to the value in DQODT during reads and disabling it (setting it to '0' during any other cycle. If this bit is not set, then the ODT control of DQ SSTL I/Os is always set to the value in DQODT.
9	RW	0x1	DQSRTT DQS Dynamic RTT Control Indicates, if set, that the ODT control of DQS SSTL I/Os be dynamically controlled by setting it to the value in DQSODT during reads and disabling it (setting it to '0' during any other cycle. If this bit is not set, then the ODT control of DQS SSTL I/Os is always set to the value in DQSODT field.
8:7	RW	0x1	DSEN Write DQS Enable Controls whether the write DQS going to the SDRAM is enabled (toggling) or disabled (static value) and whether the DQS is inverted. DQS# is always the inversion of DQS. These values are valid only when DQS/DQS# output enable is on, otherwise the DQS/DQS# is tristated. Valid settings are: 2'b00: DQS disabled (Driven to constant 0) 2'b01: DQS toggling with inverted polarity 2'b10: DQS toggling with normal polarity (This should be the default setting) 2'b11: DQS disabled (Driven to constant 1)
6	RW	0x0	DQSRPD DQSR Power Down Powers down, if set, the PDQSR cell. This bit is ORed with the common PDR configuration bit.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	DXPDR Data Power Down Receiver Powers down, when set, the input receiver on I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the common PDR configuration bit.
4	RW	0x0	DXPDD Data Power Down Driver Powers down, when set, the output driver on I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the common PDD configuration bit.
3	RW	0x0	DXIOM Data I/O Mode Selects SSTL mode (when set to 0) or CMOS mode (when set to 1) of the I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the IOM configuration bit of the individual DATX8.
2	RW	0x0	DQODT Data On-Die Termination Enables, when set, the on-die termination on the I/O for DQ and DM pins of the byte. This bit is ORed with the common DATX8 ODT configuration bit.
1	RW	0x0	DQSODT DQS On-Die Termination Enables, when set, the on-die termination on the I/O for DQS/DQS# pin of the byte. This bit is ORed with the common DATX8 ODT configuration bit.
0	RW	0x1	DXEN Data Byte Enable Enables if set the DATX8 and SSTL I/Os used on the data byte. Setting this bit to '0' disables the byte, i.e. the byte SSTL I/Os are put in power-down mode and the DLL in the DATX8 is put in bypass mode.

**DDR\_PUBL\_DX2GSR0**

Address: Operational Base + offset (0x0244)

DATX8 2 General Status Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:13	RO	0x000	DTPASS DQS Gate Training Pass Count The number of passing configurations during DQS gate training. Bits [2:0] are for rank 0, bits [5:3] for rank 1, and so on.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RO	0x0	reserved
11:8	RO	0x0	DTIERR DQS Gate Training Intermittent Error If set, indicates that there was an intermittent error during DQS gate training of the byte, such as a pass was followed by a fail then followed by another pass. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
7:4	RO	0x0	DTERR DQS Gate Training Error If set, indicates that a valid DQS gating window could not be found during DQS gate training of the byte. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
3:0	RO	0x0	DTDONE Data Training Done Indicates, if set, that the byte has finished doing data training. Bit [0] is for rank 0, bit 1 for rank 1, and so on.

**DDR\_PUBL\_DX2GSR1**

Address: Operational Base + offset (0x0248)

DATX8 2 General Status Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	RVPASS Read Valid Training Pass Count The number of passing configurations during read valid training. Bits [2:0] are for rank 0, bits [5:3] for rank 1, and so on.
19:16	RO	0x0	RVIERR Read Valid Training Intermittent Error If set, indicates that there was an intermittent error during read valid training of the byte, such as a pass was followed by a fail then followed by another pass. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
15:12	RO	0x0	RVERR Read Valid Training Error If set, indicates that a valid read valid placement could not be found during read valid training of the byte. Bit [0] is for rank 0, bit 1 for rank 1, and so on.

Bit	Attr	Reset Value	Description
11:4	RO	0x00	DQSDFT DQS Drift Used to report the drift on the read data strobe of the data byte. Valid settings are: 2'b00: No drift 2'b01: 90 deg drift 2'b10: 180 deg drift 2'b11: 270 deg drift or more Bits [1:0] are for rank 0, bits [3:2] for rank 1, and so on.
3:0	RO	0x0	DFTERR DQS Drift Error If set, indicates that the byte read data strobe has drifted by more than or equal to the drift limit set in the PHY General Configuration Register (PGCR). Bit [0] is for rank 0, bit 1 for rank 1, and so on.

**DDR\_PUBL\_DX2DLLCR**

Address: Operational Base + offset (0x024c)

DATX8 2 DLL Control Register

Bit	Attr	Reset Value	Description
31	RW	0x0	DLLDIS DLL Disable A disabled DLL is bypassed. Default ('0') is DLL enabled
30	RW	0x1	DLLSRST DLL Soft Rest Soft resets the byte DLL by driving the DLL soft reset pin
29:20	RO	0x0	reserved
19	RW	0x0	SDLBMODE Slave DLL Loopback Mode If this bit is set, the slave DLL is put in loopback mode in which there is no 90 degrees phase shift on read DQS/DQS#. This bit must be set when operating the byte PHYs in loopback mode such as during BIST loopback. Applicable only to PHYs that have this feature. Refer to PHY databook.
18	RW	0x0	ATESTEN Analog Test Enable Enables the analog test signal to be output on the DLL analog test output (test_out_a). The DLL analog test output is tri-stated when this bit is '0'.

Bit	Attr	Reset Value	Description
17:14	RW	0x0	<p>SDPHASE Slave DLL Phase Trim Selects the phase difference between the input clock and the corresponding output clock of the slave DLL. Valid settings:</p> <p>4'b0000: 90 4'b0001: 72 4'b0010: 54 4'b0011: 36 4'b0100: 108 4'b0101: 90 4'b0110: 72 4'b0111: 54 4'b1000: 126 4'b1001: 108 4'b1010: 90 4'b1011: 72 4'b1100: 144 4'b1101: 126 4'b1110: 108 4'b1111: 90</p>
13:12	RW	0x0	<p>SSTART Slave Auto Start-Up Used to control how the slave DLL starts up relative to the master DLL locking:</p> <p>2'b0X: Slave DLL automatically starts up once the master DLL has achieved lock 2'b10: The automatic startup of the slave DLL is disabled; the phase detector is disabled 2'b11: The automatic startup of the slave DLL is disabled; the phase detector is enabled</p>
11:9	RW	0x0	<p>MFWDLY Master Feed-Forward Delay Trim Used to trim the delay in the master DLL feed-forward path:</p> <p>3'b000: minimum delay 3'b111: maximum delay</p>
8:6	RW	0x0	<p>MFBFDLY Master Feed-Back Delay Trim Used to trim the delay in the master DLL feedback path:</p> <p>3'b000: minimum delay 3'b111: maximum delay</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:3	RW	0x0	SFWDLY Slave Feed-Forward Delay Trim Used to trim the delay in the slave DLL feed-forward path: 3'b000: minimum delay 3'b111: maximum delay
2:0	RW	0x0	SFBDLY Slave Feed-Back Delay Trim Used to trim the delay in the slave DLL feedback path: 3'b000: minimum delay 3'b111: maximum delay

**DDR\_PUBL\_DX2DQTR**

Address: Operational Base + offset (0x0250)

DATX8 2 DQ Timing Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RW	0xf	DQDLY7 DQ7 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte. The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are: 2'b00: nominal delay 2'b01: nominal delay + 1 step 2'b10: nominal delay + 2 steps 2'b11: nominal delay + 3 steps

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27:24	RW	0xf	<p>DQDLY6 DQ6 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
23:20	RW	0xf	<p>DQDLY5 DQ5 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19:16	RW	0xf	<p>DQDLY4 DQ4 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
15:12	RW	0xf	<p>DQDLY3 DQ3 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:8	RW	0xf	<p>DQDLY2 DQ2 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
7:4	RW	0xf	<p>DQDLY1 DQ1 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

Bit	Attr	Reset Value	Description
3:0	RW	0xf	<p>DQDLY0 DQ0 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

**DDR\_PUBL\_DX2DQSTR**

Address: Operational Base + offset (0x0254)

DATX8 2 DQS Timing Register

Bit	Attr	Reset Value	Description
31:30	RO	0x0	reserved
29:26	RW	0xf	<p>DMDLY DM Delay Used to adjust the delay of the data mask relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. The lower two bits of the DQMDLY controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b.</p> <p>Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
25:23	RW	0x3	<p>DQSNDLY DQS# Delay Used to adjust the delay of the data strobes relative to the nominal delay that is matched to the delay of the data bit through the slave DLL and clock tree. DQSDLY control the delay on DQS strobe and DQSNDLY control the delay on DQS#. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: nominal delay - 3 steps</li> <li>3'b001: nominal delay - 2 steps</li> <li>3'b010: nominal delay - 1 step</li> <li>3'b011: nominal delay</li> <li>3'b100: nominal delay + 1 step</li> <li>3'b101: nominal delay + 2 steps</li> <li>3'b110: nominal delay + 3 steps</li> <li>3'b111: nominal delay + 4 steps</li> </ul>
22:20	RW	0x3	<p>DQSDLY DQS Delay Used to adjust the delay of the data strobes relative to the nominal delay that is matched to the delay of the data bit through the slave DLL and clock tree. DQSDLY control the delay on DQS strobe and DQSNDLY control the delay on DQS#. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: nominal delay - 3 steps</li> <li>3'b001: nominal delay - 2 steps</li> <li>3'b010: nominal delay - 1 step</li> <li>3'b011: nominal delay</li> <li>3'b100: nominal delay + 1 step</li> <li>3'b101: nominal delay + 2 steps</li> <li>3'b110: nominal delay + 3 steps</li> <li>3'b111: nominal delay + 4 steps</li> </ul>
19:16	RO	0x0	reserved

Bit	Attr	Reset Value	Description
15:14	RW	0x2	<p>R1DGPS</p> <p>Rank 1 DQS Gating Phase Select</p> <p>Selects the clock used to enable the data strobes during read so that the value of the data strobes before and after the preamble/postamble are filtered out. The RnDGPS fields are initially set by the PUBL during automatic DQS data training and subsequently updated during data strobe drift compensation. However, these values can be overwritten by a direct write to this register, and the automatic update during DQS drift compensation can be disabled using the PHY General Configuration Register (PGCR). Every two bits of this register control the DQS gating for each of the (up to) four ranks. Valid values for each 2-bit RnDGPS field are:</p> <ul style="list-style-type: none"> <li>2'b00: 90 deg clock (clk90)</li> <li>2'b01: 180 deg clock (clk180)</li> <li>2'b10: 270 deg clock (clk270)</li> <li>2'b11: 360 deg clock (clk0)</li> </ul>
13:12	RW	0x2	<p>R0DGPS</p> <p>Rank 0 DQS Gating Phase Select</p> <p>Selects the clock used to enable the data strobes during read so that the value of the data strobes before and after the preamble/postamble are filtered out. The RnDGPS fields are initially set by the PUBL during automatic DQS data training and subsequently updated during data strobe drift compensation. However, these values can be overwritten by a direct write to this register, and the automatic update during DQS drift compensation can be disabled using the PHY General Configuration Register (PGCR). Every two bits of this register control the DQS gating for each of the (up to) four ranks. Valid values for each 2-bit RnDGPS field are:</p> <ul style="list-style-type: none"> <li>2'b00: 90 deg clock (clk90)</li> <li>2'b01: 180 deg clock (clk180)</li> <li>2'b10: 270 deg clock (clk270)</li> <li>2'b11: 360 deg clock (clk0)</li> </ul>
11:6	RO	0x0	reserved

Bit	Attr	Reset Value	Description
5:3	RW	0x0	<p>R1DGSL Rank 1 DQS Gating System Latency Used to increase the number of clock cycles needed to expect valid DDR read data by up to five extra clock cycles. This is used to compensate for board delays and other system delays. Power-up default is 000 (i.e. no extra clock cycles required). The SL fields are initially set by the PUBL during automatic DQS data training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each of the (up to) four ranks. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: No extra clock cycles</li> <li>3'b001: 1 extra clock cycle</li> <li>3'b010: 2 extra clock cycles</li> <li>3'b011: 3 extra clock cycles</li> <li>3'b100: 4 extra clock cycles</li> <li>3'b101: 5 extra clock cycles</li> <li>3'b110: Reserved</li> <li>3'b111: Reserved</li> </ul>
2:0	RW	0x0	<p>R0DGSL Rank 0 DQS Gating System Latency Used to increase the number of clock cycles needed to expect valid DDR read data by up to five extra clock cycles. This is used to compensate for board delays and other system delays. Power-up default is 000 (i.e. no extra clock cycles required). The SL fields are initially set by the PUBL during automatic DQS data training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each of the (up to) four ranks. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: No extra clock cycles</li> <li>3'b001: 1 extra clock cycle</li> <li>3'b010: 2 extra clock cycles</li> <li>3'b011: 3 extra clock cycles</li> <li>3'b100: 4 extra clock cycles</li> <li>3'b101: 5 extra clock cycles</li> <li>3'b110: Reserved</li> <li>3'b111: Reserved</li> </ul>

**DDR\_PUBL\_DX3GCR**

Address: Operational Base + offset (0x0280)  
DATX8 3 General Configuration Register

Bit	Attr	Reset Value	Description
31:20	RO	0x0	reserved
16:14 19:17	RW	0x3 0x3	<p>R0RVSL R1RVSL</p> <p>Rank n ITMD Read Valid System Latency Used to specify the read valid system latency relative to the ideal placement of the ITMD read valid signal when DXCCR.RVSEL is set to 0.</p> <p>Power-up default is 011 (i.e. ideal placement of the read valid signal). The RVSL fields are initially set by the PUB during automatic read valid training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each rank. R0RVSL controls the latency of rank 0, R1RVSL controls rank 1. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: read valid system latency = ideal placement - 3</li> <li>3'b001: read valid system latency = ideal placement - 2</li> <li>3'b010: read valid system latency = ideal placement - 1</li> <li>3'b011: read valid system latency = ideal placement</li> <li>3'b100: read valid system latency = ideal placement + 1</li> <li>3'b101: read valid system latency = ideal placement + 2</li> <li>3'b110: read valid system latency = ideal placement + 3</li> <li>3'b111: Reserved</li> </ul>
13	RW	0x0	<p>RTTOAL</p> <p>RTT On Additive Latency</p> <p>Indicates when the ODT control of DQ/DQS SSTL I/Os is set to the value in DQODT/DQSODT during read cycles. Valid values are:</p> <ul style="list-style-type: none"> <li>1'b0: ODT control is set to DQSODT/DQODT almost two cycles before read data preamble</li> <li>1'b1: ODT control is set to DQSODT/DQODT almost one cycle before read data preamble</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12:11	RW	0x0	RTTOH RTT Output Hold Indicates the number of clock cycles (from 0 to 3) after the read data postamble for which ODT control should remain set to DQSODT for DQS or DQODT for DQ/DM before disabling it (setting it to '0' when using dynamic ODT control. ODT is disabled almost RTTOH clock cycles after the read postamble.
10	RW	0x1	DQRTT DQ Dynamic RTT Control Indicates, if set, that the ODT control of DQ/DM SSTL I/Os be dynamically controlled by setting it to the value in DQODT during reads and disabling it (setting it to '0' during any other cycle. If this bit is not set, then the ODT control of DQ SSTL I/Os is always set to the value in DQODT.
9	RW	0x1	DQSRTT DQS Dynamic RTT Control Indicates, if set, that the ODT control of DQS SSTL I/Os be dynamically controlled by setting it to the value in DQSODT during reads and disabling it (setting it to '0' during any other cycle. If this bit is not set, then the ODT control of DQS SSTL I/Os is always set to the value in DQSODT field.
8:7	RW	0x1	DSEN Write DQS Enable Controls whether the write DQS going to the SDRAM is enabled (toggling) or disabled (static value) and whether the DQS is inverted. DQS# is always the inversion of DQS. These values are valid only when DQS/DQS# output enable is on, otherwise the DQS/DQS# is tristated. Valid settings are: 2'b00: DQS disabled (Driven to constant 0) 2'b01: DQS toggling with inverted polarity 2'b10: DQS toggling with normal polarity (This should be the default setting) 2'b11: DQS disabled (Driven to constant 1)
6	RW	0x0	DQSRPD DQSR Power Down Powers down, if set, the PDQSR cell. This bit is ORed with the common PDR configuration bit.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	DXPDR Data Power Down Receiver Powers down, when set, the input receiver on I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the common PDR configuration bit.
4	RW	0x0	DXPDD Data Power Down Driver Powers down, when set, the output driver on I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the common PDD configuration bit.
3	RW	0x0	DXIOM Data I/O Mode Selects SSTL mode (when set to 0) or CMOS mode (when set to 1) of the I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the IOM configuration bit of the individual DATX8.
2	RW	0x0	DQODT Data On-Die Termination Enables, when set, the on-die termination on the I/O for DQ and DM pins of the byte. This bit is ORed with the common DATX8 ODT configuration bit.
1	RW	0x0	DQSODT DQS On-Die Termination Enables, when set, the on-die termination on the I/O for DQS/DQS# pin of the byte. This bit is ORed with the common DATX8 ODT configuration bit.
0	RW	0x1	DXEN Data Byte Enable Enables if set the DATX8 and SSTL I/Os used on the data byte. Setting this bit to '0' disables the byte, i.e. the byte SSTL I/Os are put in power-down mode and the DLL in the DATX8 is put in bypass mode.

**DDR\_PUBL\_DX3GSR0**

Address: Operational Base + offset (0x0284)

DATX8 3 General Status Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:13	RO	0x000	DTPASS DQS Gate Training Pass Count The number of passing configurations during DQS gate training. Bits [2:0] are for rank 0, bits [5:3] for rank 1, and so on.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RO	0x0	reserved
11:8	RO	0x0	DTIERR DQS Gate Training Intermittent Error If set, indicates that there was an intermittent error during DQS gate training of the byte, such as a pass was followed by a fail then followed by another pass. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
7:4	RO	0x0	DTERR DQS Gate Training Error If set, indicates that a valid DQS gating window could not be found during DQS gate training of the byte. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
3:0	RO	0x0	DTDONE Data Training Done Indicates, if set, that the byte has finished doing data training. Bit [0] is for rank 0, bit 1 for rank 1, and so on.

**DDR\_PUBL\_DX3GSR1**

Address: Operational Base + offset (0x0288)

DATX8 3 General Status Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	RVPASS Read Valid Training Pass Count The number of passing configurations during read valid training. Bits [2:0] are for rank 0, bits [5:3] for rank 1, and so on.
19:16	RO	0x0	RVIERR Read Valid Training Intermittent Error If set, indicates that there was an intermittent error during read valid training of the byte, such as a pass was followed by a fail then followed by another pass. Bit [0] is for rank 0, bit 1 for rank 1, and so on.
15:12	RO	0x0	RVERR Read Valid Training Error If set, indicates that a valid read valid placement could not be found during read valid training of the byte. Bit [0] is for rank 0, bit 1 for rank 1, and so on.

Bit	Attr	Reset Value	Description
11:4	RO	0x00	DQSDFT DQS Drift Used to report the drift on the read data strobe of the data byte. Valid settings are: 2'b00: No drift 2'b01: 90 deg drift 2'b10: 180 deg drift 2'b11: 270 deg drift or more Bits [1:0] are for rank 0, bits [3:2] for rank 1, and so on.
3:0	RO	0x0	DFTERR DQS Drift Error If set, indicates that the byte read data strobe has drifted by more than or equal to the drift limit set in the PHY General Configuration Register (PGCR). Bit [0] is for rank 0, bit 1 for rank 1, and so on.

**DDR\_PUBL\_DX3DLLCR**

Address: Operational Base + offset (0x028c)

DATX8 3 DLL Control Register

Bit	Attr	Reset Value	Description
31	RW	0x0	DLLDIS DLL Disable A disabled DLL is bypassed. Default ('0') is DLL enabled
30	RW	0x1	DLLSRST DLL Soft Rest Soft resets the byte DLL by driving the DLL soft reset pin
29:20	RO	0x0	reserved
19	RW	0x0	SDLBMODE Slave DLL Loopback Mode If this bit is set, the slave DLL is put in loopback mode in which there is no 90 degrees phase shift on read DQS/DQS#. This bit must be set when operating the byte PHYs in loopback mode such as during BIST loopback. Applicable only to PHYs that have this feature. Refer to PHY databook.
18	RW	0x0	ATESTEN Analog Test Enable Enables the analog test signal to be output on the DLL analog test output (test_out_a). The DLL analog test output is tri-stated when this bit is '0'.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17:14	RW	0x0	<p>SDPHASE Slave DLL Phase Trim Selects the phase difference between the input clock and the corresponding output clock of the slave DLL. Valid settings:</p> <p>4'b0000: 90 4'b0001: 72 4'b0010: 54 4'b0011: 36 4'b0100: 108 4'b0101: 90 4'b0110: 72 4'b0111: 54 4'b1000: 126 4'b1001: 108 4'b1010: 90 4'b1011: 72 4'b1100: 144 4'b1101: 126 4'b1110: 108 4'b1111: 90</p>
13:12	RW	0x0	<p>SSTART Slave Auto Start-Up Used to control how the slave DLL starts up relative to the master DLL locking:</p> <p>2'b0X: Slave DLL automatically starts up once the master DLL has achieved lock 2'b10: The automatic startup of the slave DLL is disabled; the phase detector is disabled 2'b11: The automatic startup of the slave DLL is disabled; the phase detector is enabled</p>
11:9	RW	0x0	<p>MFWDLY Master Feed-Forward Delay Trim Used to trim the delay in the master DLL feed-forward path:</p> <p>3'b000: minimum delay 3'b111: maximum delay</p>
8:6	RW	0x0	<p>MFBFDLY Master Feed-Back Delay Trim Used to trim the delay in the master DLL feedback path:</p> <p>3'b000: minimum delay 3'b111: maximum delay</p>

Bit	Attr	Reset Value	Description
5:3	RW	0x0	SFWDLY Slave Feed-Forward Delay Trim Used to trim the delay in the slave DLL feed-forward path: 3'b000: minimum delay 3'b111: maximum delay
2:0	RW	0x0	SFBDLY Slave Feed-Back Delay Trim Used to trim the delay in the slave DLL feedback path: 3'b000: minimum delay 3'b111: maximum delay

**DDR\_PUBL\_DX3DQTR**

Address: Operational Base + offset (0x0290)

DATX8 3 DQ Timing Register

Bit	Attr	Reset Value	Description
31:28	RW	0xf	DQDLY7 DQ7 Delay DQ Delay: Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte. Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte. The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are: 2'b00: nominal delay 2'b01: nominal delay + 1 step 2'b10: nominal delay + 2 steps 2'b11: nominal delay + 3 steps

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27:24	RW	0xf	<p>DQDLY6 DQ6 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
23:20	RW	0xf	<p>DQDLY5 DQ5 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19:16	RW	0xf	<p>DQDLY4 DQ4 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
15:12	RW	0xf	<p>DQDLY3 DQ3 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:8	RW	0xf	<p>DQDLY2 DQ2 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>
7:4	RW	0xf	<p>DQDLY1 DQ1 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

Bit	Attr	Reset Value	Description
3:0	RW	0xf	<p>DQDLY0 DQ0 Delay Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. Every four bits of this register control the delay of a different data bit in the byte.</p> <p>The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b. Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

**DDR\_PUBL\_DX3DQSTR**

Address: Operational Base + offset (0x0294)

DATX8 3 DQS Timing Register

Bit	Attr	Reset Value	Description
31:30	RO	0x0	reserved
29:26	RW	0xf	<p>DMDLY DM Delay Used to adjust the delay of the data mask relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree. The lower two bits of the DQMDLY controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b.</p> <p>Valid settings for each 2-bit control field are:</p> <ul style="list-style-type: none"> <li>2'b00: nominal delay</li> <li>2'b01: nominal delay + 1 step</li> <li>2'b10: nominal delay + 2 steps</li> <li>2'b11: nominal delay + 3 steps</li> </ul>

Bit	Attr	Reset Value	Description
25:23	RW	0x3	<p>DQSNDLY DQS# Delay Used to adjust the delay of the data strobes relative to the nominal delay that is matched to the delay of the data bit through the slave DLL and clock tree. DQSDLY control the delay on DQS strobe and DQSNDLY control the delay on DQS#. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: nominal delay - 3 steps</li> <li>3'b001: nominal delay - 2 steps</li> <li>3'b010: nominal delay - 1 step</li> <li>3'b011: nominal delay</li> <li>3'b100: nominal delay + 1 step</li> <li>3'b101: nominal delay + 2 steps</li> <li>3'b110: nominal delay + 3 steps</li> <li>3'b111: nominal delay + 4 steps</li> </ul>
22:20	RW	0x3	<p>DQSDLY DQS Delay Used to adjust the delay of the data strobes relative to the nominal delay that is matched to the delay of the data bit through the slave DLL and clock tree. DQSDLY control the delay on DQS strobe and DQSNDLY control the delay on DQS#. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: nominal delay - 3 steps</li> <li>3'b001: nominal delay - 2 steps</li> <li>3'b010: nominal delay - 1 step</li> <li>3'b011: nominal delay</li> <li>3'b100: nominal delay + 1 step</li> <li>3'b101: nominal delay + 2 steps</li> <li>3'b110: nominal delay + 3 steps</li> <li>3'b111: nominal delay + 4 steps</li> </ul>
19:16	RO	0x0	reserved

Bit	Attr	Reset Value	Description
15:14	RW	0x2	<p>R1DGPS</p> <p>Rank 1 DQS Gating Phase Select</p> <p>Selects the clock used to enable the data strobes during read so that the value of the data strobes before and after the preamble/postamble are filtered out. The RnDGPS fields are initially set by the PUBL during automatic DQS data training and subsequently updated during data strobe drift compensation. However, these values can be overwritten by a direct write to this register, and the automatic update during DQS drift compensation can be disabled using the PHY General Configuration Register (PGCR). Every two bits of this register control the DQS gating for each of the (up to) four ranks. Valid values for each 2-bit RnDGPS field are:</p> <ul style="list-style-type: none"> <li>2'b00: 90 deg clock (clk90)</li> <li>2'b01: 180 deg clock (clk180)</li> <li>2'b10: 270 deg clock (clk270)</li> <li>2'b11: 360 deg clock (clk0)</li> </ul>
13:12	RW	0x2	<p>R0DGPS</p> <p>Rank 0 DQS Gating Phase Select</p> <p>Selects the clock used to enable the data strobes during read so that the value of the data strobes before and after the preamble/postamble are filtered out. The RnDGPS fields are initially set by the PUBL during automatic DQS data training and subsequently updated during data strobe drift compensation. However, these values can be overwritten by a direct write to this register, and the automatic update during DQS drift compensation can be disabled using the PHY General Configuration Register (PGCR). Every two bits of this register control the DQS gating for each of the (up to) four ranks. Valid values for each 2-bit RnDGPS field are:</p> <ul style="list-style-type: none"> <li>2'b00: 90 deg clock (clk90)</li> <li>2'b01: 180 deg clock (clk180)</li> <li>2'b10: 270 deg clock (clk270)</li> <li>2'b11: 360 deg clock (clk0)</li> </ul>
11:6	RO	0x0	reserved

Bit	Attr	Reset Value	Description
5:3	RW	0x0	<p>R1DGSL</p> <p>Rank 1 DQS Gating System Latency</p> <p>Used to increase the number of clock cycles needed to expect valid DDR read data by up to five extra clock cycles. This is used to compensate for board delays and other system delays. Power-up default is 000 (i.e. no extra clock cycles required). The SL fields are initially set by the PUBL during automatic DQS data training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each of the (up to) four ranks. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: No extra clock cycles</li> <li>3'b001: 1 extra clock cycle</li> <li>3'b010: 2 extra clock cycles</li> <li>3'b011: 3 extra clock cycles</li> <li>3'b100: 4 extra clock cycles</li> <li>3'b101: 5 extra clock cycles</li> <li>3'b110: Reserved</li> <li>3'b111: Reserved</li> </ul>
2:0	RW	0x0	<p>R0DGSL</p> <p>Rank 0 DQS Gating System Latency</p> <p>Used to increase the number of clock cycles needed to expect valid DDR read data by up to five extra clock cycles. This is used to compensate for board delays and other system delays. Power-up default is 000 (i.e. no extra clock cycles required). The SL fields are initially set by the PUBL during automatic DQS data training but these values can be overwritten by a direct write to this register. Every three bits of this register control the latency of each of the (up to) four ranks. Valid values are:</p> <ul style="list-style-type: none"> <li>3'b000: No extra clock cycles</li> <li>3'b001: 1 extra clock cycle</li> <li>3'b010: 2 extra clock cycles</li> <li>3'b011: 3 extra clock cycles</li> <li>3'b100: 4 extra clock cycles</li> <li>3'b101: 5 extra clock cycles</li> <li>3'b110: Reserved</li> <li>3'b111: Reserved</li> </ul>

## 13.6 Interface description

DDR IOs for each channel are listed as following Table.

Pin Name	Description	
CK	Active-high clock signal to the memory device.	13.7 A
CK_B	Active-low clock signal to the memory device.	ppli
CKE <sub>i</sub> (i=0,1)	Active-high clock enable signal to the memory device for two chip select.	cati
CS_B <sub>i</sub> (i=0,1)	Active-low chip select signal to the memory device. There are two chip select.	on
RAS_B	Active-low row address strobe to the memory device.	Not
CAS_B	Active-low column address strobe to the memory device.	es
WE_B	Active-low write enable strobe to the memory device.	13.7.1 S
BA[2:0]	Bank address signal to the memory device.	tate
A[15:0]	Address signal to the memory device.	tran
DQ[31:0]	Bidirectional data line to the memory device.	
DQS[3:0]	Active-high bidirectional data strobes to the memory device.	
DQS_B[3:0]	Active-low bidirectional data strobes to the memory device.	
DM[3:0]	Active-low data mask signal to the memory device.	
ODT <sub>i</sub> (i=0,1)	On-Die Termination output signal for two chip select.	
RET_EN	Active-low retention latch enable input.	
VREF <sub>i</sub> (i=0,1,2)	Reference Voltage input for three regions of DDR IO.	
ZQ_PIN	ZQ calibration pad which connects 240ohm±1% resistor.	
RESET	DDR3 reset signal.	

### sition of PCTL

To operate PCTL, the programmer must be familiar with the available operational states and how to transition to each state from the current state.

Every software programmable register is accessible only during certain operational states. For information about what registers are accessible in each state, refer to "Software Registers," which provides this information in each register description. The general rule is that the PCTL must be in the Init\_mem or Config states to successfully write most of the registers.

The following tables provide the programming sequences for moving to the various states of the state machine.

### Moving to the Init\_mem State

Step	Application	PCTL
1	Read STAT register	Returns the current PCTL state.
2	If STAT.ctrl_stat = Init_mem, go to END.	
3	If STAT.ctrl_stat =Config, go to Step9.	
4	If STAT.ctrl_stat =Access, go to Step8.	
5	If STAT.ctrl_stat = Low_power, go to Step7.	
6	Goto Step1.	PCTL is in a Transitional state and not in any of the previous operational states.
7	Write WAKEUP to SCTL.state_cmd and poll STAT.ctrl_stat= Access.	Issues SRX, moves to the Access state, updates STAT.ctrl_stat =Access when complete.
8	Write CFG to SCTL.state_cmd and poll STAT.ctrl_stat= Config.	PCTL stalls the NIF; completes any pending transaction; issues PREA if required; moves into the Config state; updates STAT.ctrl_stat =Config when complete.

9	Write INIT to <b>SCTL.state_cmd</b> and poll <b>STAT.ctl_stat=Init_mem</b>	Moves into the Init_mem state and updates <b>STAT.ctl_stat=Init_mem</b> .
END		PCTL is in Init_mem state.

### Moving to Config State

Step	Application	PCTL
1	Read <b>STAT</b> register.	Returns the current PCTL state.
2	If <b>STAT.ctl_stat= Config</b> , goto END.	
3	If <b>STAT.ctl_stat= Low_power</b> , go to <b>Step6</b> .	
4	If <b>STAT.ctl_stat= Init_mem</b> or <b>Access</b> , go to <b>Step7</b> .	
5	Go to <b>Step1</b> .	PCTL is in a transitional state and is not in any of the previous operational states.
6	Write WAKEUP to <b>CTL.state_cmd</b> and poll <b>STAT.ctl_stat= Access</b> .	Issues SRX, moves to the Access state, and updates <b>STAT.ctl_stat= Access</b> when complete.
7	Write CFG to <b>SCTL.state_cmd</b> and poll <b>STAT.ctl_stat= Config</b> .	PCTL stalls the NIF; completes any pending transaction; issues PREA if required; moves into the Config state; and updates <b>STAT.ctl_stat = Config</b> when complete.
END		PCTL is in Config state.

### Moving to Access State

Step	Application	PCTL
1	Read <b>STAT</b> register	Returns the current PCTL state.
2	If <b>STAT.ctl_stat= Access</b> , go to END.	
3	If <b>STAT.ctl_stat= Config</b> , go to <b>Step9</b>	
4	If <b>STAT.ctl_stat= Init_mem</b> , go to <b>Step8</b>	
5	If <b>STAT.ctl_stat= Low_power</b> , go to <b>Step7</b> .	
6	Goto <b>Step1</b> .	PCTL is in a transitional state and is not in any of the previous operational states.
7	Write WAKEUP to <b>SCTL.state_cmd</b> and poll <b>STAT.ctl_stat= Access</b> . Goto END	Issues SRX, moves to the Access state, updates <b>STAT.ctl_stat= Access</b> when complete.
8	Write CFG to <b>SCTL.state_cmd</b> and poll <b>STAT.ctl_stat= Config</b> .	Moves into the Config state, updates <b>STAT.ctl_stat= Config</b> when complete.
9	Write GO to <b>SCTL.state_cmd</b> and poll <b>STAT.ctl_stat= Access</b> .	Moves into the Access state, updates <b>STAT.ctl_stat= Access</b> when complete.
END		PCTL is in Access state.

### Moving to Low Power State

Step	Application	PCTL
1	Read <b>STAT</b> register.	Returns current PCTL state.
2	If <b>STAT.ctl_stat =Low_power</b> , go to END.	

3	If <b>STAT.ctl_stat</b> = Access, go to <b>Step9</b>	
4	If <b>STAT.ctl_stat</b> = Config, go to <b>Step8</b>	
5	If <b>STAT.ctl_stat</b> = Init_mem, go to <b>Step7</b> .	
6	Goto <b>Step1</b> .	PCTL is in transitional state and is not in any of the previous operational states.
7	Write CFG to <b>SCTL.state_cmd</b> and poll <b>STAT.ctl_stat</b> = Config.	Moves into the Config state, updates <b>STAT.ctl_stat</b> = Config when complete.
8	Write GO to <b>SCTL.state_cmd</b> and poll <b>STAT.ctl_stat</b> = Access.	Moves into the Access state, updates <b>STAT.ctl_stat</b> =Access when complete.
9	Write SLEEP to <b>SCTL.state_cmd</b> and poll <b>STAT.ctl_stat</b> = Low_power.	Issues PDX if necessary; completes any pending transactions; issues PREA command; finally, issues SRE and updates <b>STAT.ctl_stat</b> = Low_power.
END		PCTL is in Low Power state

### 13.7.2 Initialization

Figure 1-14 shows a high-level illustration of the initialization sequence of the PHY. A detailed sequence description and timing diagrams are described in the following. This section assumes a generic configuration port and therefore **cfg\_clk** and **cfg\_rst\_n** are shown as the configuration clock and reset, respectively. These signals must be replaced by **pclk** and **presetn** if the design is compiled to use the APB configuration port.

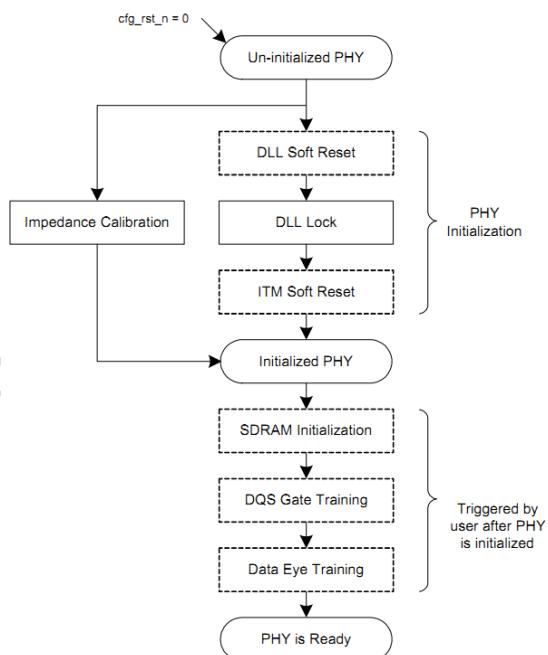


Fig. 13-10 Protocol controller architecture

#### PHY Initialization

The initialization sequence has two phases. The first phase happens automatically at reset and is as follows:

1. Before and during configuration reset (i.e. if **cfg\_rst\_n** is asserted), the PHY is un-initialized and remains in this state until the reset is de-asserted.

2. At reset de-assertion, the PHY moves into the DLL initialization (lock) phase. This phase may be bypassed at any time by writing a '1' to the DLL initialization bypass register bit (PIR[LOCKBYP]).

3. In parallel to DLL initialization, the impedance calibration phase also starts at reset de-assertion.

This phase can also be bypassed by writing a '1' to the impedance calibration bypass register bit-PIR[ZCALBYP].

4. If the PHY initialization sequence was triggered by the user, a soft reset may optionally be selected to be issued to the ITMs. Initialization that is automatically triggered on reset does not issue a soft reset to the ITMs because the components will already have been reset by the main reset.

5. Once the DLL initialization and impedance calibration phases are done and after the ITMs are reset, the PHY is initialized. Note that if these phases were bypassed, it is up to the user to perform them in software or trigger them at a later time before the PHY can be used.

## **SDRAM Initialization**

The second phase of initialization starts after the PHY is initialized. Each step of this phase is triggered by the user or memory controller and is as follows:

1. Prior to normal operation, DDR SDRAMs must be initialized. The PHYCTL has a built-in SDRAM initialization routine that may be triggered by software or memory controller by writing to the PHY Initialization Register (PIR). The initialization routine built into the PHYCTL is generic and does not require any knowledge of the type or configuration of external SDRAMs to be properly executed. The routine is designed with the relevant JEDEC specifications for the fastest & slowest SDRAMs supported by the PHYCTL to result in a universal initialization sequence. This generic sequence is applicable to DDR3, DDR2, LPDDR2, LPDDR, and DDR SDRAMs.

It is recommended to use the built-in PHYCTL routine to initialize the SDRAM. However, there may be cases such as during system debug when the built-in PHYCTL DRAM initialization is not triggered and DRAM initialization is performed by software or the controller. In these cases the system must first wait for the PHY to initialize, i.e. DLL locked and impedance calibration done, then it must write a '1' to PIR[INIT] bit with PIR[CTLDINT] set to '1' (for controller initialization) or '0' (for software or PHYCTL initialization) to inform the PHYCTL that DRAM initialization will be done later, by software, the controller or by re-triggering on the PHYCTL. The software or controller then executes the initialization sequence by sending relevant commands to the DRAM, respecting the various timing requirements of the initialization sequence.

2. After the SDRAM is initialized, the user or memory controller performs, or triggers the PHYCTL to perform DQS gate training ("Built-in DQS Gate Training" on page 114). The SDRAM must be initialized before triggering DQS gate training.

3. The user or memory controller performs, or triggers the PHYCTL to perform read data eye training. Note that the current version of the PHYCTL does not have the read eye training designed in.

4. The PHY is now ready for SDRAM read/write accesses.

## **DDR3 Initialization Sequence**

The initialization steps for DDR3 SDRAMs are as follows:

1. Optionally maintain RESET# low for a minimum of either 200 us (power-up initialization) or 100ns (power-on initialization). The PHYCTL drives RESET# low from the beginning of reset assertion and therefore this step may be skipped when DRAM initialization is triggered if

enough time may already have expired to satisfy the RESET# low time.

2. After RESET# is de-asserted, wait a minimum of 500 us with CKE low.
3. Apply NOP and drive CKE high.
4. Wait a minimum of tXPR.
5. Issue a load Mode Register 2 (MR2) command.
6. Issue a load Mode Register 3 (MR3) command.
7. Issue a load Mode Register (MR1) command (to set parameters and enable DLL).
8. Issue a load Mode Register (MR0) command to set parameters and reset DLL.
9. Issue ZQ calibration command.
10. Wait 512 SDRAM clock cycles for the DLL to lock (tDLLK) and ZQ calibration (tZQinit) to finish. This wait time is relative to Step 8, i.e. relative to when the DLL reset command was issued onto the SDRAM command bus.

### **LPDDR2 Initialization Sequence**

The initialization steps for LPDDR2 SDRAMs are as follows:

1. Wait a minimum of 100 ns (tINIT1) with CKE driven low.
2. Apply NOP and set CKE high.
3. Wait a minimum of 200 us (tINIT3).
4. Issue a RESET command.
5. Wait a minimum of 1 us + 10 us (tINIT4 + tINIT5).
6. Issue a ZQ calibration command.
7. Wait a minimum of 1 us (tZQINIT).
8. Issue a Write Mode Register to MR1.
9. Issue a Write Mode Register to MR2
10. Issue a Write Mode Register to MR3

### **Initialization Triggered and bypass**

All initialization steps shown in Figure 3-1 on page 34 can be triggered using the PHY Initialization Register (PIR) as described in “PHY Initialization Register (PIR)” on page 47. Writing a '1' to PIR[INIT] register bit will start initialization, with the routines to be run being selected by the corresponding PIR register bits. If multiple routines are selected, they are run in the order shown in Figure 3-1 on page 34. This is also the order of the select bits in PIR register. The completion of the routines is indicated in the PHY General Status Register (PGSR) with the corresponding done status bits (see “PHY General Status Register (PGSR)” on page 52). The PGSR[IDONE] bit indicates the overall completion of the initialization sequence. An initialization done status register bit is cleared (reset to '0') when the corresponding routine is re-triggered.

The de-assertion of reset will automatically trigger the PHYCTL to perform DLL initialization (locking) and impedance calibration. Once the DLL has locked and impedance calibration has completed, the SDRAM initialization and DQS gating may be triggered or performed by software or memory controller.

Since the PHYCTL allows the selection of individual routines to be run when initialization is triggered using PIR register, only those routines that automatically trigger on reset de-assertion have individual bypass capability. This means that DLL locking and/or impedance calibration may be bypassed any time by writing a '1' to the corresponding bypass register bit in the PIR register. Once a routine is bypassed, it is internally registered as completed and the corresponding done status register bit is set in the PGSR register.

It is up to the user to re-trigger or perform the bypassed routine at a later time before the PHY can be used. The PIR[INITBYP] register bit provides the option to bypass the whole initialization sequence.

### 13.7.3 MDLL and MSDLL Reset Requirements

Reset issued to the MDLL and MSDLL must always meet the following requirements:

1. Reset must always be asserted for a minimum of 50ns to ensure proper reset of the DLL.
2. On power-up, reset must be held for a minimum of 50ns after MVDD has been raised to its full value.
3. After reset has been asserted and then de-asserted, a number of clock cycles must pass for the DLL to achieve lock.
4. The input clock to the DLL must be stable for a minimum of 50ns before DLL reset is de-asserted.

The following additional requirements apply when transitioning to/from bypass mode:

1. There must be at least 50ns between reset de-assertion and DLL bypass mode entry.
2. The DLL bypass pin must be asserted for at least 1000ns.
3. Reset must always be issued after the DLL mode has changed from bypass to normal mode.
4. A minimum of 100ns is required between bypass de-assertion and reset assertion.
5. Reset must be issued whenever DLL control/trim/option input bits are modified, with the exception of:

- a. Analog/digital test controls
- b. Slave DLL phase trim (if applicable).

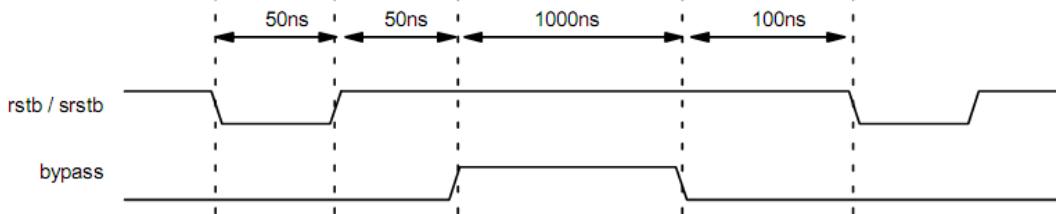


Fig. 13-11 DLL reset requirements

### 13.7.4 Data Training

#### Built-in DQS Gate Training

The PHYCTL has a built-in DQS gate training routine that may be triggered by software or memory controller using the PIR register.

DQS gate training returns a number of status, including the done and error status. There are two types of errors. The first type is when no valid window was found for the byte. This is indicated by DTERR register bit in DXnGSR and PGSR registers. This is usually an indication of

bad configuration. The second type is when some passing configurations were found but these were interspersed by failures. This is not expected in a working system. A typical window is signified by consecutive passes followed by consecutive failures, e.g. FFFFFFFF and not FFFF. This type of error is called an intermittent error and is indicated by the

DTIERR register bit in DXnGSR and PGSR registers. Provided for debug purpose is the status of how many passing configurations were found for each byte on each rank. This is indicated by DTPASS field in the DXnGSR register.

### Software DQS Gate Training

DQS gate training may also be executed in software using the controller and/or the PUB DCU. Figure 13-16 shows the DQS gate training software algorithm. This is followed by a description of the main phases of the training.

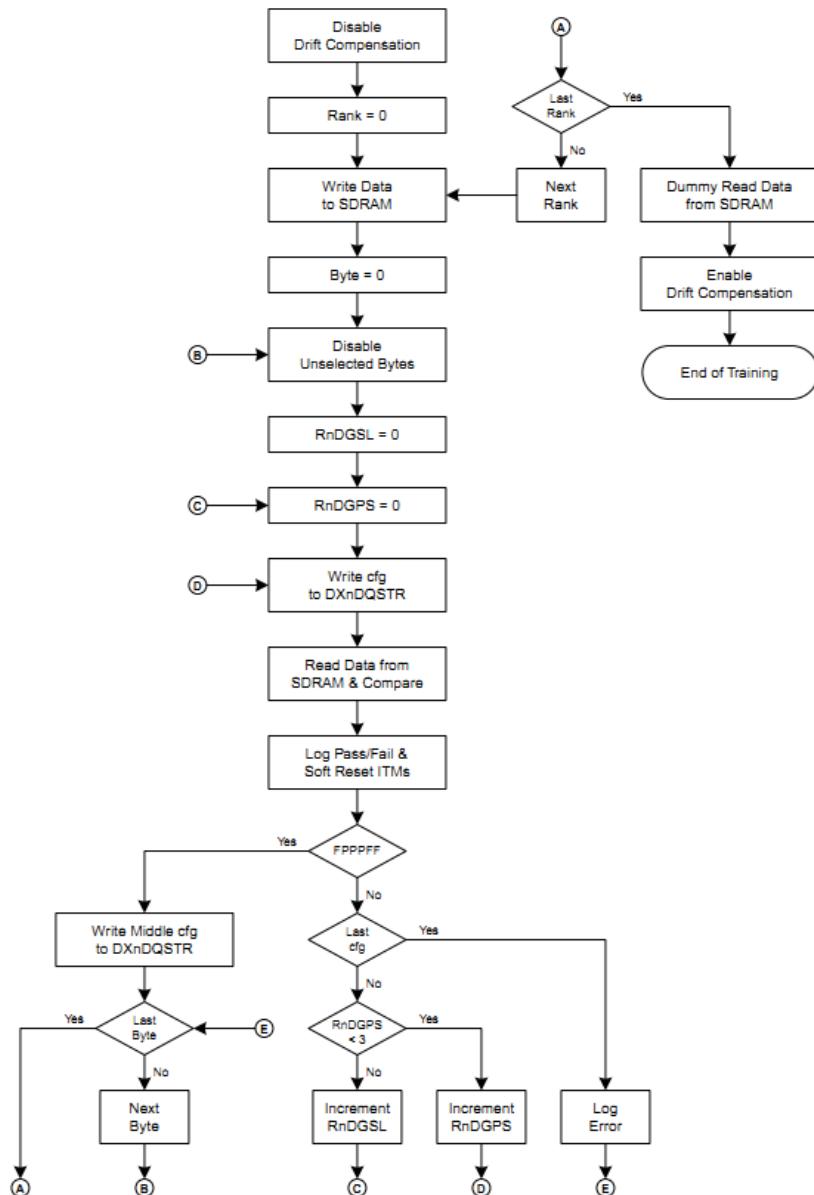


Fig. 13-12 DLL reset requirements

The software DQS gate training phases are as follows:

1. Disable drift compensation by writing '0' to PGCR.DFTCMP register.
2. Start with rank 0, i.e. rank 0 is selected for training.

3. Execute a minimum of two writes to the SDRAM. Any type of data and any SDRAM address can be used for DQS gate training. It is however not recommended to use data that is all zeroes since this may mask read data comparison. The data mask must be set to 0 to enable writing of all bytes. The number of writes must be chosen such that it results in a minimum of eight data beats at the SDRAM. This means at least two write commands when using SDRAM burst length of 4.

4. Start with byte 0 (i.e. byte 0 is selected for training).

5. Disable all the other bytes except the byte that has been selected for training. Bytes are enabled/disabled by writing 1/0 to DXnDGCR.DXEN.

6. Start with the selected rank byte DQS gating system latency (DXnDQSTR.RnDGSL) of 0.

7. Start with the selected rank byte DQS gating phase select (DXnDQSTR.RnDGPS) of 0.

8. Write the selected DQS gating configurations (RnDGSL and RnDGPS) to DXnDQSTR register of the selected byte, making sure the fields for the unselected ranks remain unchanged.

9. Execute reads from the SDRAM locations previously written. The number of reads must be equal to the number of writes used in Step 3. Compare the read data with the expected (written) data and log the pass/fail status as a sequence or history of flags for each trained RnDGSL/RnDGPS configuration (e.g. FFPPPPFF). A fail is either when there is a data miscompare or when fewer data than expected is returned. Note that a controller that is designed to always wait for the correct number of read data may need a time-out in case the trained configuration results in fewer data than expected. This is not an issue when using the PUB DCU because it does not wait for the expected number of reads; rather the read count status will indicate if fewer reads were returned.

10. Once the read data has been compared and the pass/fail status logged, issue an ITM soft reset to clear the status of the read data logic in the PHY. This is important because the ITM read data FIFO pointers may be in the wrong state at the end of training an RnDGSL/RnDGPS configuration that resulted in wrong DQS gating window.

11. If two consecutive fails and some passes exist, then this is the end of the training for this rank byte. In this case, do the following:

- Select the middle of the passes and write the values to the corresponding fields of DXnDQSTR register, making sure the fields for the unselected ranks remain unchanged
- If this is not the last byte, then select the next byte and go to Step 5
- If this is the last byte but not the last rank, then select the next rank and go to Step 3
- If this is the last byte and the last rank, then go to Step 12 to do final clean-up before the end of the DQS gate training.
- If the condition of two consecutive fails and some passes does not exist, then this signals that more RnDGSL/RnDGPS configurations need to be trained for this rank byte. If this is the case, do the following:
  - if RnDGPS is less than 3, then increment RnDGPS and go to Step 8
  - if RnDGPS is equal to 3 but RnDGSL is less than 7, then increment RnDGSL and go to Step 7
  - if RnDGPS is equal to 3 and RnDGSL is equal to 7, then log an error because this is a signal that something in the system is very wrong such that no passing configuration is possible for this rank byte. With such an error condition, you can either terminate the whole training to investigate the system or you can go to train the next byte.

12. Once the training of all ranks and all bytes is finished, issue one or more dummy reads to the same SDRAM locations. This will flush out the DQS drift compensation logic in the PHY and therefore avoid reporting any false drift events caused by previous DQS gating settings.

13. Once the dummy reads have completed, re-enable drift compensation by writing 1 to PGCR.DFTCMP register. This is the end of DQS gate training. Regular memory operations can now commence.

### 13.7.5 Impedance Calibration

The impedance calibration circuit, which controls the impedance values for ODT and driver output impedance, consists of the following components:

- ZQ calibration cell - PZQ
- External RZQ precision resistor
- Impedance control logic - zctrl
- VREF cell (for code encoding and level shifting)
- Functional I/O cells

The connectivity of these components is shown as follow figure:

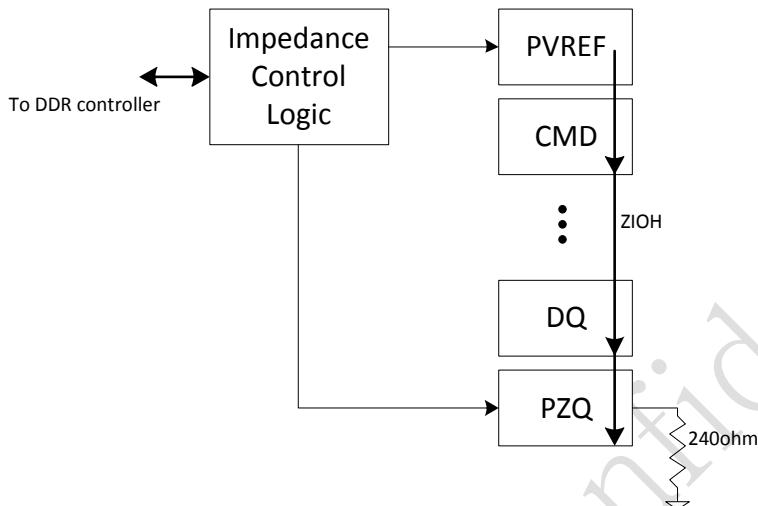


Fig. 13-13 Impedance Calibration Circuit

A single calibration cell (PZQ) is used for the interface. The user connects the PZQ pin through an external  $240\text{ohm}\pm1\%$  resistor to ground. One or multiple VREF cells exist in the interface, depending on the total data width of the interface. The ZCTRL bus from the impedance control logic is connected to all VREF cells in the interface. It is not permitted to have a VREF cell in the interface that is not connected to the impedance control logic.

The impedance control logic sends an impedance code through the ZCTRL bus to the VREF cells. The VREF cells encodes this data, level shifts it to the VDDQ power domain, and sends it to both the functional I/O cells and the PZQ cell through the ZIOH bus embedded within the SSTL cells. The PZQ cell also receives the desired divide ratios from the Memory Controller or the user logic. The PZQ cell compares the impedance control code received from the PVREF cell with the external resistor, taking into account the selected divide ratio. The PZQ cell then sends ZCOMP back to the impedance control logic to relay information about impedance matching. The impedance control logic then sends a new impedance code to the PVREF cells. This results in a closed-loop system.

The four impedance elements are calibrated sequentially:

- Pull-up termination impedance
- Pull-down termination impedance
- Pull-up output impedance
- Pull-down output impedance

The ZPROG bus is used to signal which element is being calibrated. The state machine is implemented on the Impedance Controller RTL block.

The impedance control logic connects to the Memory Controller or customer logic to allow full controllability and observability of the loop operation.

The impedance control loop operates with a low bandwidth as compared to the memory system, thus the impedance control logic contains a clock divider to permit operation at a

reduced clock frequency.

There are three basic modes of operation:

- Direct Calibration - uses ZPROG settings.
- Override Setting - uses ctrl\_ovrd\_data settings.
- Custom Calibration - extends calibration beyond the values available on ZPROG

### **Direct Calibration**

In this mode, the user is setting independently the value for ODT (ZPROG[7:4]) and Output Impedance (ZPROG[3:0]) and runs the calibration sequence:

1. Output impedance pulldown
2. Output impedance pull-up
3. On-Die termination (ODT) pull-down
4. ODT pull-up

### **Override Setting**

In this mode, the user is not using the calibration loop, and instead directly controls the impedance control using zctrl\_ovrd\_data[19:0] bus, which is parsed in four nibbles that independently control driver pull-down/up and ODT pull-down/up impedance in 31 steps.

For example, assuming one step is associated to current I and the calibration voltage is VREF, the programmed impedance for index N is:

$$\text{ZPROG} = K * \text{VREF}/(\text{N} * \text{I})$$

K is correction factor, which is approximately equal to 1. Based on the formula, it can be concluded that if index N is increased, then the impedance is decreased.

### **Custom Calibration**

This mode is a two-step procedure combining the previous two modes.

1. The user provides a Direct Calibration using a convenient value and records the Impedance control results from status register.
2. The user applies the correction factor that provides the custom impedance.

The following example assumes that it is required to program Driver Output Impedance to 18 ohms.

1. The user performs a Direct Calibration for driver  $Z_o=36$  ohms. For example, assume the result shows that Driver pull-up index is 12, and Driver pull-down index is 13.
2. Calculate and apply the Override Data for 18 ohm impedance adjustment as follows:  
 $(\text{<cal_value>} / \text{<req_value>}) * \text{<cal_index>}$   
Driver pull-down  $(36/18) * 13 = 26$   
Driver pull-up  $(36/18) * 12 = 24$

### **13.7.6 Retention Functional**

The purpose of the retention function is to retain a known state on the signals to the SDRAMs while the system is placed in a low power mode, specifically when the core VDD supply is powered down. The general concept is that an external input signal (RET\_EN) is driven low to put the SSTL I/O cells into retention mode shortly before the core VDD supply is powered down. The user must set the SSTL I/O outputs in the state required during power down before asserting RET\_EN. This ensures that the output state of all SSTL I/Os are held static in the desired state while core VDD is power down. After core VDD is restored, the user must

re-initialize the core logic to a known state before de-asserting the RET\_EN signal.

Following figure provides the I/O cell arrangement with retention.

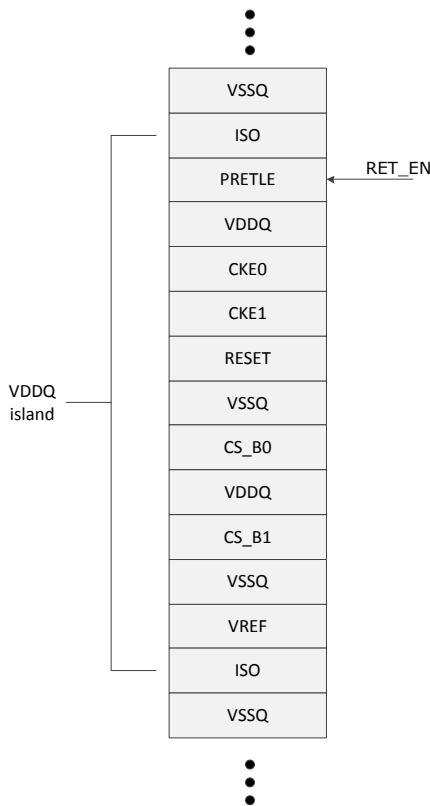


Fig. 13-14 I/O cell arrangement with retention

IOs between two ISO is a VDDQ island, they will maintain power on when other IOs are powered down by RET\_EN active.

Following figure provides a sequence of events to enter and exit retention.

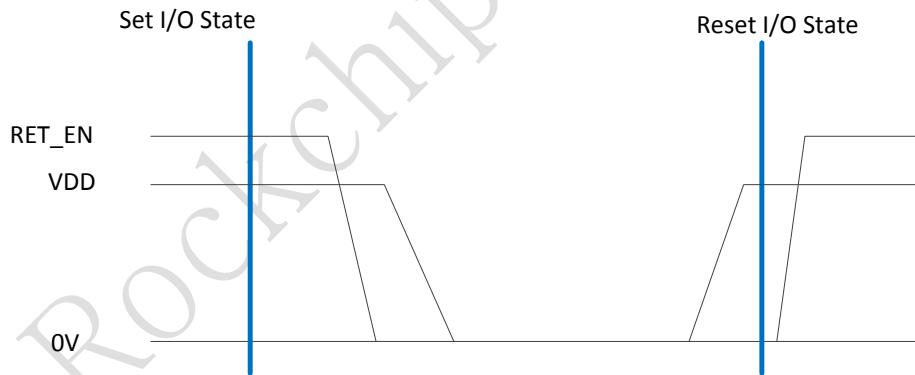


Fig. 13-15 Sequence of Events to Enter and Exit Retention

### CKE Retention Mode

An alternative CKE retention mode is supported. This scheme works by placing the SDRAMs into self-refresh mode and then driving the CKE signal low. Core VDD and VDDQ can then both be powered down except for a small VDDQ island supplying the CKE output cell. Two of the special 5um spacer cells ISO are used to break the VDDQ rail in order to create a separate CKE VDDQ island, which is kept powered while core VDD and the main VDDQ are powered down.

The sequence of events is as follows:

1. Enter self-refresh mode using the Self-Refresh Command
2. Set CKE low
3. Stop CK/CKB

4. Assert RET\_EN (low)
5. Power-Off
6. Power-On
7. After reset is released, execute initialization
8. De-assert RET\_EN (high)
9. Start CK/CKB
10. Set CKE high
11. Exit self-refresh mode

### 13.7.7 Low Power Operation

Low\_power state can be entered/exited via following ways:

- Software control of PCTL State machine (highest priority)
- Hardware Low Power Interface (middle priority)
- Auto Self Refresh feature (lowest priority)

Note the priority of requests from Access to Low\_power is highlighted above. The STAT.ip\_trig register field reports which of the 3 requests caused the entry to Low\_power state.

#### Software control of PCTL State

The application can request via software to enter the memories into Self Refresh state by issuing the SLEEP command by programming SCTL.PCTL responds to the software request by moving into the Low\_power operational state and issuing the SRE command to the memories. Note that the Low\_power state can only be reached from the Access state.

In a similar fashion, the application requests to exit the memories from Self Refresh by issuing a WAKEUP command by programming SCTL.. PCTL responds to the WAKEUP command issuing SRX and restoring normal NIF address channel operation.

#### Hardware Low Power Interface

The hardware low power interface can also be used to enter/exit Self Refresh. The functionality is enabled by setting SCFG.hw\_low\_power\_en=1. Once that bit is set, the input c\_sysreq has the ability to trigger entry into the Low Power configuration state just like the software methodology (SCTL.state\_cmd = SLEEP). A hardware Low Power entry trigger will be ignored/denied if the input c\_active\_in=1 or n\_valid=1. It may be accepted if c\_active\_in=0 and n\_valid=0, depending on the current state of the PCTL. When SCFG.hw\_low\_power\_en=1, the outputs c\_sysack and c\_active provide feedback as required by the AXI low power interface specification (this interface's operation is defined by the AXI specification). c\_sysack acknowledges the request to go into the Low\_power state, and c\_active indicates when the PCTL is actually in the Low\_power state.

The c\_active output could also be used by an external Low Power controller to decide when to request a transition to low power. When MCFG1.hw\_idle > 0, c\_active = 1'b0 indicates that the NIF has been idle for at least MCFG1.hw\_idle \* 32 \* n\_clk cycles while in the Access state.

When in low power the c\_active output can be used by an external Low Power controller to trigger a low power exit. c\_active will be driven high when either c\_active\_in or n\_valid are high. The path from c\_active\_in and n\_valid to c\_active is asynchronous so even if the clocks have been removed c\_active will assert. The Low Power controller should re-enable the clocks when c\_active is driven high while in the Low\_power state.

#### Auto Clock Stop/Power Down/Self Refresh

The Clock Stop and/or Power Down and/or Self Refresh sequence is automatically started by PCTL when the NIF address channel is idle for a number of cycles, depending on the programmed value in MCFG.mddr\_lpddr2\_clkstop\_idle and MCFG.pd\_idle and MCFG1.sr\_idle.

Following table outlines the effect of these settings in conjunction with NIF being idle.

<b>mddr_lpddr2_clkstop_idle</b>	<b>pd_idle</b>	<b>sr_idle</b>	<b>Memory modes</b>	<b>Memory Type</b>
0	0	0	none	All
>0	0	0	Clock Stop	mDDR/LPDDR2 only
0	>0	0	Power Down	All
>0	>0	0	Clock Stop -> Power Down <sup>①</sup>	mDDR/LPDDR2 only
0	0	>0	Self Refresh	All
>0	0	>0	Clock Stop -> Self Refresh <sup>②</sup>	mDDR/LPDDR2 only
0	>0	>0	Power Down -> Self Refresh <sup>③</sup>	All
>0	>0	>0	Clock Stop -> Power Down -> Self Refresh <sup>④</sup>	mDDR/LPDDR2 only

Note:

①: Clock Stop is entered if NIF is idle for mddr\_lpddr2\_clkstop\_idle. Following on from that, if NIF continues to be idle for a further pd\_idle cycles, Clock Stop is exited and Power Down is entered.

②: Clock Stop is entered if NIF is idle for mddr\_lpddr2\_clkstop\_idle. Following on from that, if NIF continues to be idle for a further sr\_idle\*32 cycles, Clock Stop is exited and Self Refresh is entered.

③: Power Down is entered if NIF is idle for pd\_idle. Following on from that, if NIF continues to be idle for a further sr\_idle\*32 cycles, Power Down is exited and Self Refresh is entered.

④: Clock Stop is entered if NIF is idle for mddr\_lpddr2\_clkstop\_idle. Following on from that, if NIF continues to be idle for a further pd\_idle cycles, Clock Stop is exited and Power Down is entered. Following on from that, if NIF continues to be idle for a further sr\_idle\*32 cycles, Power Down is exited and Self Refresh is entered.

## Removing PCTL's n\_clk

In LPDDR2 and DDR3, the relationship between SRE/SRX and stopping/starting the memory clock (CK) are formalized and are accounted for automatically by PCTL. With LPDDR2 and DDR3, CK should only be stopped after PCTL has reached the Low\_power state. The current operational state can be verified by reading STAT.ctl\_stat. The CK must be started and stable before the Software or Hardware Low Power Interface attempts to take the memory out of Self Refresh.

PCTL's n\_clk can be safely removed when PCTL is in Low Power state. The sequences outlined in following table should be followed for safe operation:

<b>Step</b>	<b>Application</b>	<b>PCTL</b>
1	Write SLEEP to SCTL.state_cmd and poll STAT.ctl_stat = LOW_POWER.	Tells PCTL to move memories into Self Refresh and waits until this completes.
2	Write TREFI=0. Also, write DFITCRLUPI=0 and DFIREFMSKI=0, if they are not already 0.	Stops any MC-driven DFI updates occurring internally with PCTL
3	Wait a minimum interval which is equivalent to the PCTL's Refresh Interval (previous value of TREFI*TOGCNT100N*internal timers clock period;	Ensures any already scheduled PHY/PVT updates have completed successfully.
4	Stop toggling n_clk to PCTL.	n_clk logic inside PCTL is stopped.
end		

Step	Application	PCTL
1	Drive c_active_in low	Confirms that system external to PCTL can accept a Low-power request
2	Drive c_sysreq low	System Low-power request
3	Wait for PCTL to drive c_sysack low	PCTL Low-power request acknowledgement
4	Check value of c_active when Step 3 occurs. - if c_active=1, request denied. Cannot remove n_clk. Go to END. - if c_active=0, request accepted.	PCTL low-power request status response
5	Stop toggling n_clk to PCTL	n_clk logic inside PCTL is stopped
end		

### Deep Power-Down

Compared with DDR2/DDR3, mDDR and LPDDR2 has an additional low power mode (Deep Power Down) :

- Software-driven Deep Power Down Entry – on reception of DPDE from the application, PCTL drives CKE low for TDPD.t\_dpd. After TDPD, MCMD.start\_cmd will be cleared to 1'b0. The following are recommended values for TDPD:
  - mDDR: TDPD=0
  - LPDDR2: dependent on if the system wants to immediately power off the PCTL after Deep Power down is entered:
    - If PCTL not Powered off: TDPD=500μs
    - Else if PCTL is Powered off: TDPD=0 - up to higher level system to meet tDPD requirement.
- To Exit Deep Power Mode, full initialization of the memories must be performed.

### 13.7.8 PHY Power Down

The PHYCTL includes several registers for putting certain components of the PHY in power down mode. The PHTCTL also supports DFI-initiated power-down of its components using the DFI low-power protocol.

Several components of the PHY can be powered down using PHYCTL registers. There are separate power-down register bits for the address/command lane and for each byte lane. Also there are separate controls for powering down the I/Os versus powering down the DLL. Following table describes the registers that are used to power down various components of the PHY.

Register Name	Bit Field	Description
PIR	DLLBYP	Bypasses, and hence disables or powers down all PHY DLLs.
ACDLLCR	DLLDIS	Disables (powers down) the address/command lane DLL
ACIOCR	ACPDD	Powers down the output drivers for address/command lane signal I/Os. Different groups of signals have dedicated driver power-down control registers to allow finer selection of signals to power down, especially that some signals, such as CKE and RST#, are required to remain powered up when the SDRAM is in self-refresh mode. Each rank CS# signal and each CK/CK# pair has dedicated driver power down control registers, with the other rank-specific signals (CKE and ODT) of each rank being

		controlled by separate power down control registers in a separate PUB register (DSGCR). There is also a dedicated driver power down control register for SDRAM reset signal. However, the rest of the signals going to the SDRAM (address, bank address, RAS#, CAS#, WE#, and PAR_IN) share a common driver power down register just dedicated for this group. The LPDDR TPD signal has a dedicated output driver power down control register in a separate PUB register (DSGCR).
ACIOCR	ACPDR	Powers down the input receivers for address/command lane signal I/Os. Different groups of signals have dedicated receiver power-down control registers to allow finer selection of signals to power down. Each rank and each CK/CK# pair has dedicated receiver power down control register, with all rank-specific signals (CKE, ODT, and CS#) of each rank sharing a common, but rank-specific, receiver power down control register. There is also a dedicated receiver power down control register for SDRAM reset pins. However, the rest of the signals going to the SDRAM (address, bank address, RAS#, CAS#, WE#, PAR_IN, TPD) share a common receiver power down register just dedicated for this group.
DXCCR	DXPDD	Powers down the output drivers for DQ, DM, and DQS/DQS# signal I/Os of all byte lanes. This is a convenient way of powering down the output drivers of all byte lane I/Os with just a single register write. In addition to this, each byte has a dedicated output driver power-down register control to allow only selected bytes to be powered down.
DXCCR	DXPDR	Powers down the input receivers for DQ, DM, and DQS/DQS# signal I/Os of all byte lanes. It also powers down the PDQSR cells of all bytes. This is a convenient way of powering down the input receivers of all byte lane I/Os with just a single register write. In addition to this, each byte has a dedicated input receiver power-down register control to allow only selected bytes to be powered down.
DSGCR	CKEPDD	Powers down the output drivers for CKE I/Os. Each rank CKE has a dedicated driver power down control register to allow finer control of CKE I/O driver power-down, especially that the CKE I/O driver of an SDRAM that is in self refresh is required to remain powered up.
DSGCR	ODTPDD	Powers down the output drivers for ODT I/Os. Each rank ODT has a dedicated driver power down control register to allow finer control of ODT I/O driver power-down, especially that the ODT I/O driver of an SDRAM that is in self refresh or power down mode may be required in certain DDR modes to remain powered up.
DSGCR	TPDPD	Powers down the output driver for the optional LPDDR TPD signal I/O.
DSGCR	NL2PD	Powers down the output driver and the input receiver on the I/O for non-LPDDR2 signals (ODT, RAS#, CAS#, WE#, and BA). This may be used when a chip that is designed for both LPDDR2 and other DDR modes is being used in LPDDR2 mode, in which case one may want to power down the unused I/Os. This power down control register is in addition to (ORed with) the individual ACIOCR power down control registers for these signals.
ZQnCRO	ZQPD	Powers down the PZQ cell. Each PZQ has a dedicated power down control register.
DXnDLLCR	DLLDIS	Disables (powers down) the byte lane DLL. Each byte lane has a dedicated DLL power down control register.
DXnGCR	DXPDD	Powers down the output drivers for DQ, DM, and DQS/DQS# signal I/Os of the byte lane. Each byte lane has a dedicated output driver power down control register, in conjunction with the global output driver power down control register DXCCR.DXPDD.

DXnGCR	DXPDR	Powers down the input receivers for DQ, DM, and DQS/DQS# signal I/Os of the byte lane. Each byte lane has a dedicated input receiver power down control register, in conjunction with the global input receiver power down control register DXCCR.DXPDR
DXnGCR	DQSRPD	Powers down the PDQSR cells of the byte lane. Each byte lane has a dedicated PDQSR power down control register, in conjunction with the global PDQSR power down control register DXCCR.DXPDR.
PGCR	PDDISDX	Selects whether the I/Os and DLL of a disabled byte should automatically be powered down by the PUB. A byte can be disabled by writing a '0' to the DXnGCR.DXEN register or by using the DFI data byte disable (dfi_data_byte_disable) signal.
DSGCR	LPIOPD	Specifies whether the PHY should respond to the controller-initiated DFI low power opportunity request and power down the I/Os of the PHY.
DSGCR	LPDLLPD	Specifies whether the PHY should respond to the controller-initiated DFI low power opportunity request and power down the DLL of the PHY if the requested wakeup time is greater than 2048 clock cycles

### DFI-Initiated Power-Down

There are two ways how the controller can initiate PHY power down through the DFI interface. The first method is when the controller asserts the DFI data byte disable (dfi\_data\_byte\_disable) signal during initialization when the DFI initialization start (dfi\_init\_start) signal is high. In this state, the PHY will power down the DLL and I/Os of the selected bytes if it is configured through DSGCR.BDISEN to respond to DFI data byte disable and if disabled bytes are configured through PGCR.PDDISDX to be powered down. The DFI data byte disable feature is normally used as a static configuration to disable bytes that are not being used.

The controller can also initiate PHY power down by using the DFI low power control interface. This is a dynamic low power request-acknowledge protocol that the controller may use to put the PHY into low power mode when it is not being used for a prolonged time. The PHY will acknowledge a low power request from the controller and power down I/Os and DLLs if it is configured to do so through DSGCR.LPIOPD and DSGCR.LPDLLPD. If the low power wakeup time requested by the controller is less than 2048 clock cycles, then only the I/Os will be powered down. Otherwise if the wakeup time is equal to or more than 2048 cycles, then the DLLs and the I/Os are all powered down. If the DLLs are powered down, then on low power wakeup the PUB will soft reset the DLLs and wait for them to lock before acknowledging the low power wakeup request to the controller.

### 13.7.9 Dynamic ODT for I/Os

By default the DFI turns on the ODT for the PHY I/Os for DQ/DQS# only when there is read data coming back. This is called dynamic ODT control and is used to reduce power consumed by the termination resistors. The DFI uses the timing of the DQS gating to accurately place the PHY I/O ODT enable signal around the read data. Typically, the DFI turns on the byte ODT enable signal 2 clocks before the pre-amble and turns it off one clock after the post-amble. This guarantees correct setup and hold on the I/Os.

The PHY ODT signal does not go through the ITMs and therefore has to fan out to the DQ/DQS from RTL logic in the PHYCTL. This may result in different timing on these signals depending on the routing. For this reason various programmable features are provided on the ODT control signals to help mitigate some of the timing issues that may result from different implementations. These are described in the DXnGCR register. In summary, both the starting position and the width of the enable signal can be adjusted relative to the default position and

lengths.

Rockchip Confidential

## Chapter 14 Mobile Storage Host Controller

### 14.1 Overview

The Mobile Storage Host Controller is designed to support Secure Digital memory (SD- max version 3.01) with 1 bits or 4 bits data width, Multimedia Card(MMC-max version 4.51) with 1 bits or 4 bits or 8 bits data width.

The Host Controller supports following features:

- Bus Interface Features:
  - Supports AMBA AHB interface for master and slave
  - Supports internal DMA interface(IDMAC)
    - ◆ Supports 16/32-bit data transfers
    - ◆ Single-channel; single engine used for Transmit and Receive, which are mutually exclusive
    - ◆ Dual-buffer and chained descriptor linked list
    - ◆ Each descriptor can transfer up to 4KB of data in chained mode and 8KB of data in dual-buffer mode
    - ◆ Programmable burst size for optimal host bus utilization
  - Supports combined single FIFO for both transmit and receive operations
  - Supports FIFO size of 256x32
  - Supports FIFO over-run and under-run prevention by stopping card clock
- Card Interface Features:
  - Supports Secure Digital memory protocol commands
  - Supports Secure Digital I/O protocol commands
  - Supports Multimedia Card protocol commands
  - Supports Command Completion Signal and interrupts to host
  - Supports CRC generation and error detection
  - Supports programmable baud rate
  - Supports power management and power switch
  - Supports card detection and initialization
  - Supports write protection
  - Supports hardware reset
  - Supports SDIO interrupts in 1-bit and 4-bit modes
  - Supports 4-bit mode in SDIO3.0
  - Supports SDIO suspend and resume operation
  - Supports SDIO read wait
  - Supports block size of 1 to 65,535 bytes
  - Supports 1-bit, 4-bit and 8-bit SDR modes
  - Supports 4-bit DDR,8-bit DDR, as defined by SD3.0 and MMC4.41
  - Supports boot in 1-bit, 4-bit and 8-bit SDR modes
  - Supports Packed Commands, CMD21, CMD49
- Clock Interface Features:
  - Supports 0/90/180/270-degree phase shift operation for sample clock(cclk\_in\_sample) and drive clock(cclk\_in\_drv) relative to function clock(cclk\_in) respectively
  - Supports phase tuning using delay line for sample clock(cclk\_in\_sample) and drive clock(cclk\_in\_drv) relative to function clock (cclk\_in) respectively. The max number of delay element number is 256.

The Host Controller is instantiated for SDMMC, SDIO0, SDIO1, EMMC in RK3288. The interface

difference between these instances is shown in "Interface Description".

## 14.2 Block Diagram

The Host Controller consists of the following main functional blocks.

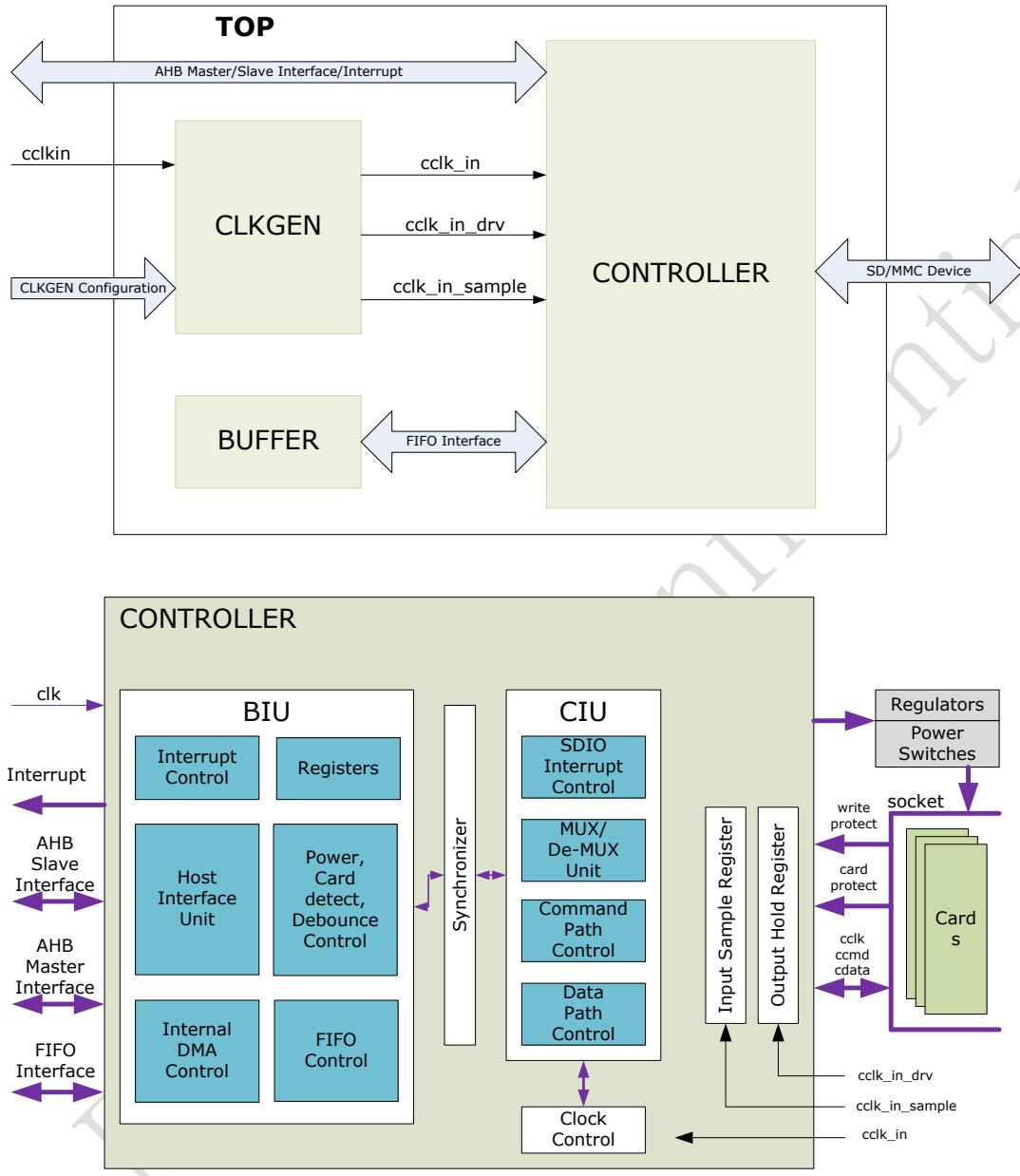


Fig. 14-1 Host Controller Block Diagram

- Clock Generate Unit(CLKGEN): generates card interface clock `cclk_in`/`cclk_sample`/`cclk_drv` based on `cclk_in` and configuration information.
- Asynchronous dual-port memory(BUFFER): Uses a two-clock synchronous read and synchronous write dual-port RAM. One of the ports is connected to the host clock, and the second port is connected to the card clock.
- Bus Interface Unit (BIU): Provides AMBA AHB interfaces for register and data read/writes.
- Card Interface Unit (CIU): Takes care of the SD/MMC protocols and provides clock management.

## 14.3 Function Description

### 14.3.1 Bus Interface Unit

The Bus Interface Unit provides the following functions:

- Host interface
- Interrupt control
- Register access
- External FIFO access
- Power control and card detection

#### 1. Host Interface Unit

The Host Interface Unit is an AHB slave interface, which provides the interface between the SD/MMC card and the host bus. You can configure the host interface as either an AHB.

#### 2. Register Unit

The register unit is part of the bus interface unit; it provides read and write access to the registers.

All registers reside in the Bus Interface Unit clock domain. When a command is sent to a card by setting the start\_bit, which is bit[31] of the CMD register, all relevant registers needed for the CIU operation are transferred to the CIU block. During this time, the registers that are transferred from the BIU to the CIU should not be written. The software should wait for the hardware to clear the start bit before writing to these registers again. The register unit has a hardware locking feature to prevent illegal writes to registers. The lock is necessary in order to avoid metastability violations, both because the host and card clock domains are different and to prevent illegal software operations.

Once a command start is issued by setting the start\_bit of the CMD register, the following registers cannot be reprogrammed until the command is accepted by the card interface unit:

- CMD – Command
- CMDARG – Command Argument
- BYTCNT – Byte Count
- BLKSIZ – Block Size
- CLKDIV – Clock Divider
- CLKENA – Clock Enable
- CLKSRC – Clock Source
- TMOUT – Timeout
- CTYPE – Card Type

The hardware resets the start\_bit once the CIU accepts the command. If a host write to any of these registers is attempted during this locked time, then the write is ignored and the hardware lock error bit is set in the raw interrupt status register. Additionally, if the interrupt is enabled and not masked for a hardware lock error, then an interrupt is sent to the host.

When the Card Interface Unit is in an idle state, it typically takes the following number of clocks for the command handshake, where clk is the BIU clock and cclk\_in is the CIU clock:

$$3 \text{ (clk)} + 3 \text{ (cclk\_in)}$$

Once a command is accepted, you can send another command to the CIU-which has a one-deep command queue-under the following conditions:

- If the previous command was not a data transfer command, the new command is sent to the SD/MMC card once the previous command completes.

- If the previous command is a data transfer command and if wait\_prvdata\_complete (bit[13]) of the Command register is set for the new command, the new command is sent to the SD/MMC card only when the data transfer completes.
- If the wait\_prvdata\_complete is 0, then the new command is sent to the SD/MMC card as soon as the previous command is sent. Typically, you should use this only to stop or abort a previous data transfer or query the card status in the middle of a data transfer.

### 3. Interrupt Controller Unit

The interrupt controller unit generates an interrupt that depends on the controller raw interrupt status, the interrupt-mask register, and the global interrupt-enable register bit. Once an interrupt condition is detected, it sets the corresponding interrupt bit in the raw interrupt status register. The raw interrupt status bit stays on until the software clears the bit by writing a 1 to the interrupt bit; a 0 leaves the bit untouched.

The interrupt port, int, is an active-high, level-sensitive interrupt. The interrupt port is active only when any bit in the raw interrupt status register is active, the corresponding interrupt mask bit is 1, and the global interrupt enable bit is 1. The interrupt port is registered in order to avoid any combinational glitches.

The int\_enable is reset to 0 on power-on, and the interrupt mask bits are set to 32'h0, which masks all the interrupts.

*Notes:*

*Before enabling the interrupt, it is always recommended that you write 32'hffff\_ffff to the raw interrupt status register in order to clear any pending unserviced interrupts. When clearing interrupts during normal operation, ensure that you clear only the interrupt bits that you serviced.*

*The SDIO Interrupts, Receive FIFO Data Request (RXDR), and Transmit FIFO Data Request (TXDR) are set by level-sensitive interrupt sources. Therefore, the interrupt source should be first cleared before you can clear the interrupt bit of the Raw Interrupt register. For example, on seeing the Receive FIFO Data Request (RXDR) interrupt, the FIFO should be emptied so that the "FIFO count greater than the RX-Watermark" condition, which triggers the interrupt, becomes inactive. The rest of the interrupts are triggered by a single clock-pulse-width source.*

Table 14-1 Bits in Interrupt Status Register

<b>Bits</b>	<b>Interrupt</b>	<b>Description</b>
24	sdio_interrupt	Interrupt from SDIO card. In MMC-Ver3.3-only mode, these bits are always 0
16	Card no-busy	If card exit busy status, the interrupt happened
15	End Bit Error (read) /Write no CRC (EBE)	Error in end-bit during read operation, or no data CRC or negative CRC received during write operation. <i>Notes: For MMC CMD19, there may be no CRC status returned by the card. Hence, EBE is set for CMD19. The application should not treat this as an error.</i>
14	Auto Command Done (ACD)	Stop/abort commands automatically sent by card unit and not initiated by host; similar to Command Done (CD) interrupt.
13	Start Bit Error (SBE)	Error in data start bit when data is read from a card. In 4-bit mode, if all data bits do not have start bit, then this error is set.
12	Hardware Locked write Error (HLE)	During hardware-lock period, write attempted to one of locked registers.
11	FIFO Underrun/Overrun Error (FRUN)	Host tried to push data when FIFO was full, or host tried to read data when FIFO was empty. Typically this should not happen, except due to error in software.

		Card unit never pushes data into FIFO when FIFO is full, and pop data when FIFO is empty.
10	Data Starvation by Host Timeout (HTO)	<p>To avoid data loss, card clock out (cclk_out) is stopped if FIFO is empty when writing to card, or FIFO is full when reading from card.</p> <p>Whenever card clock is stopped to avoid data loss, data-starvation timeout counter is started with data-timeout value. This interrupt is set if host does not fill data into FIFO during write to card, or does not read from FIFO during read from card before timeout period.</p> <p>Even after timeout, card clock stays in stopped state, with CIU state machines waiting. It is responsibility of host to push or pop data into FIFO upon interrupt, which automatically restarts cclk_out and card state machines.</p> <p>Even if host wants to send stop/abort command, it still needs to ensure it has to push or pop FIFO so that clock starts in order for stop/abort command to send on cmd signal along with data that is sent or received on data line.</p>
9	Data Read Timeout (DRTO)	Data timeout occurred. Data Transfer Over (DTO) also set if data timeout occurs.
8	Response Timeout (RTO)	Response timeout occurred. Command Done (CD) also set if response timeout occurs. If command involves data transfer and when response times out, no data transfer is attempted by Host Controller.
7	Data CRC Error (DCRC)	Received Data CRC does not match with locally-generated CRC in CIU.
6	Response CRC Error (RCRC)	Response CRC does not match with locally-generated CRC in CIU.
5	Receive FIFO Data Request (RXDR)	Interrupt set during read operation from card when FIFO level is greater than Receive-Threshold level.
4	Transmit FIFO Data Request (TXDR)	Interrupt set during write operation to card when FIFO level reaches less than or equal to Transmit-Threshold level.
3	Data Transfer Over (DTO)	<p>Data transfer completed, even if there is Start Bit Error or CRC error. This bit is also set when “read data-timeout” occurs.</p> <p><i>Notes: DTO bit is set at the end of the last data block, even if the device asserts MMC busy after the last data block.</i></p>
2	Command Done(CD)	Command sent to card and got response from card, even if Response Error or CRC error occurs. Also set when response timeout occurs
1	Response Error (RE)	Error in received response set if one of following occurs: <ul style="list-style-type: none"> <li>● Transmission bit != 0</li> <li>● Command index mismatch</li> <li>● End-bit != 1</li> </ul>
0	Card-Detect (CDT)	When card inserted or removed, this interrupt occurs. Software should read card-detect register (CDETECT, 0x50) to determine current card status.

#### 4. FIFO Controller Unit

The FIFO controller interfaces the external FIFO to the host interface and the card controller unit. When FIFO overrun and under-run conditions occur, the card clock stops in order to avoid data loss.

The FIFO uses a two-clock synchronous read and synchronous write dual-port RAM. One of the ports is connected to the host clock, clk, and the second port is connected to the card clock, cclk\_in.

*Notes: The FIFO controller does not support simultaneous read/write access from the same port. For debugging purposes, the software may try to write into the FIFO and read back the data; results are indeterminate, since the design does not support read/write access from the same port.*

#### 5. Power Control and Card Detection Unit

The register unit has registers that control the power. Power to each card can be selectively turned on or off.

The card detection unit looks for any changes in the card-detect signals for card insertion or card removal. It filters out the debounces associated with mechanical insertion or removal, and generates one interrupt to the host. You can program the debounce filter value.

On power-on, the controller should read in the card\_detect port and store the value in the memory. Upon receiving a card-detect interrupt, it should again read the card\_detect port and XOR with the previous card-detect status to find out which card has interrupted. If more than one card is simultaneously removed or inserted, there is only one card-detect interrupt; the XOR value indicates which cards have been disturbed. The memory should be updated with the new card-detect value.

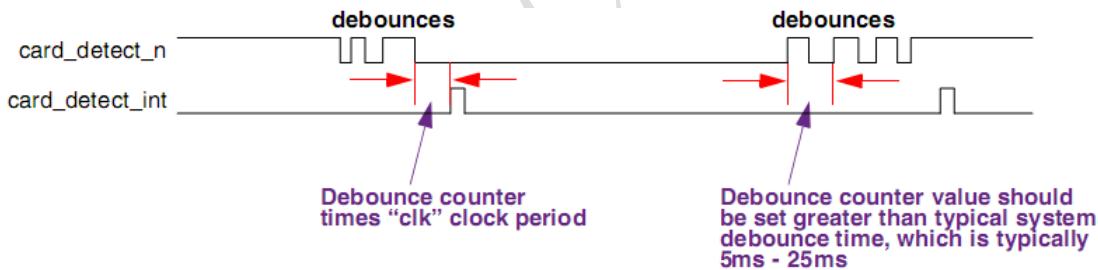


Fig. 14-2 SD/MMC Card-Detect Signal

##### 14.3.2 Card Interface Unit

The Card Interface Unit (CIU) interfaces with the Bus Interface Unit (BIU) and the devices. The host writes command parameters to the BIU control registers, and these parameters are then passed to the CIU. Depending on control register values, the CIU generates SD/MMC command and data traffic on a selected card bus according to SD/MMC protocol. The Host Controller accordingly controls the command and data path.

The following software restrictions should be met for proper CIU operation:

- Only one data transfer command can be issued at a time.
- During an open-ended card write operation, if the card clock is stopped because the FIFO is empty, the software must first fill the data into the FIFO and start the card clock. It can then issue only a stop/abort command to the card.
- When issuing card reset commands (CMD0, CMD15 or CMD52\_reset) while a card data transfer is in progress, the software must set the stop\_abort\_cmd bit in the Command register so that the Host Controller can stop the data transfer after issuing the card reset command.
- When the data end bit error is set in the RINTSTS register, the Host Controller does not guarantee SDIO

interrupts. The software should ignore the SDIO interrupts and issue the stop/abort command to the card, so that the card stops sending the read data.

- If the card clock is stopped because the FIFO is full during a card read, the software should read at least two FIFO locations to start the card clock.

The CIU block consists of the following primary functional blocks:

- Command path
- Data path
- SDIO interrupt control
- Clock control
- Mux/demux unit

## 1. Command Path

The command path performs the following functions:

- Loads clock parameters
- Loads card command parameters
- Sends commands to card bus (ccmd\_out line)
- Receives responses from card bus (ccmd\_in line)
- Sends responses to BIU
- Drives the P-bit on command line

A new command is issued to the Host Controller by programming the BIU registers and setting the start\_cmd bit in the Command register. The BIU asserts start\_cmd, which indicates that a new command is issued to the SD/MMC device. The command path loads this new command (command, command argument, timeout) and sends acknowledge to the BIU by asserting cmd\_taken.

Once the new command is loaded, the command path state machine sends a command to the device bus-including the internally generated CRC7-and receives a response, if any. The state machine then sends the received response and signals to the BIU that the command is done, and then waits for eight clocks before loading a new command.

### Load Command Parameters

One of the following commands or responses is loaded in the command path:

- New command from BIU – When start\_cmd is asserted, then the start\_cmd bit is set in the Command register.
- Internally-generated auto-stop command – When the data path ends, the stop command request is loaded.
- IRQ response with RCA 0x000 – When the command path is waiting for an IRQ response from the MMC card and a “send irq response” request is signaled by the BIU, then the send\_irq\_response bit is set in the control register.

Loading a new command from the BIU in the command path depends on the following Command register bit settings:

- update\_clock\_registers\_only – If this bit is set in the Command register, the command path updates only the clock enable, clock divider, and clock source registers. If this bit is not set, the command path loads the command, command argument, and timeout registers; it then starts processing the new command.
- wait\_prvdata\_complete – If this bit is set, the command path loads the new command under one of the following conditions:
  - Immediately, if the data path is free (that is, there is no data transfer in progress), or if an open-ended data transfer is in progress (byte\_count = 0).
  - After completion of the current data transfer, if a predefined data transfer is in progress.

### Send Command and Receive Response

Once a new command is loaded in the command path, update\_clock\_registers\_only bit is unset – the command path state machine sends out a command on the device bus; the command path state machine is illustrated in following figure.

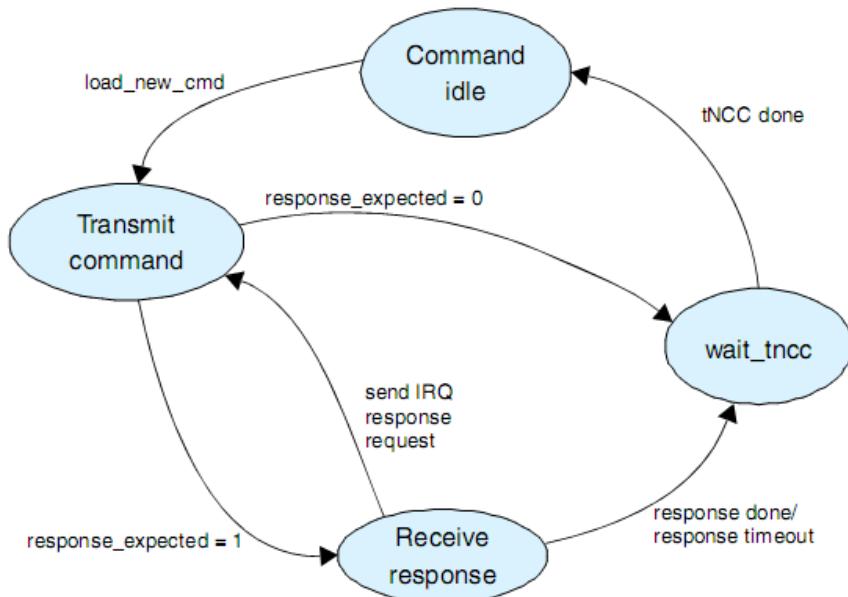


Fig. 14-3 Host Controller Command Path State Machine

The command path state machine performs the following functions, according to Command register bit values:

- send\_initialization – Initialization sequence of 80 clocks is sent before sending the command.
- response\_expected – Response is expected for the command. After the command is sent out, the command path state machine receives a 48-bit or 136-bit response and sends it to the BIU. If the start bit of the card response is not received within the number of clocks programmed in the timeout register, then the response timeout and command done bit is set in the Raw Interrupt Status register as a signal to the BIU. If the response-expected bit is not set, the command path sends out a command and signals a response done to the BIU; that is, the command done bit is set in the Raw Interrupt Status register.
- response\_length – If this bit is set, a 136-bit response is received; if it is not set, a 48-bit response is received.
- check\_response\_crc – If this bit is set, the command path compares CRC7 received in the response with the internally-generated CRC7. If the two do not match, the response CRC error is signaled to the BIU; that is, the response CRC error bit is set in the Raw Interrupt Status register.

### Send Response to BIU

If the response\_expected bit is set in the Command register, the received response is sent to the BIU. The Response0 register is updated for a short response, and the Response3, Response2, Response1, and Response0 registers are updated on a long response, after which the Command Done bit is set. If the response is for an auto\_stop command sent by the CIU, the response is saved in the Response1 register, after which the Auto Command Done bit is set.

Additionally, the command path checks for the following:

- Transmission bit = 0
- Command index matches command index of the sent command
- End bit = 1 in received card response

The command index is not checked for a 136-bit response or if the check\_response\_crc bit is unset. For a 136-bit response and reserved CRC 48-bit responses, the command index is reserved—that is, 111111.

## Polling Command Completion Signal

The device generates the Command Completion Signal in order to notify the host controller of the normal command completion or command termination.

## Command Completion Signal Detection and Interrupt to Host Processor

If the ccs\_expected bit is set in the Command register, the Command Completion Signal (CCS) from the device is indicated by setting the Data Transfer Over (DTO) bit in the RINTSTS register. The Host Controller generates a Data Transfer Over (DTO) interrupt if this interrupt is not masked.

## Command Completion Signal Timeout

If the command expects a CCS from the device—if the ccs\_expected bit is set in the Command register—the command state machine waits for the CCS and remains in a wait\_CCSS state. If the device fails to send out the CCS, the host software should implement a timeout mechanism to free the command and data path. The host controller does not implement a hardware timer; it is the responsibility of the host software to maintain a software timer.

In the event of a CCS timeout, the host should issue a CCSD by setting the send\_ccsd bit in the CTRL register. The host controller command state machine sends the CCSD to the device and exits to an idle state. After sending the CCSD, the host should also send a CMD12 to the device in order to abort the outstanding command.

## Send Command Completion Signal Disable

If the send\_ccsd bit is set in the CTRL register, the host sends a Command Completion Signal Disable (CCSD) pattern on the CMD line. The host can send the CCSD while waiting for the CCS or after a CCS timeout happens.

After sending the CCSD pattern, the host sets the Command Done (CD) bit in RINTSTS and also generates an interrupt to the host if the Command Done interrupt is not masked.

## 2. Data Path

The data path block pops the data FIFO and transmits data on cdata\_out during a write data transfer, or it receives data on cdata\_in and pushes it into the FIFO during a read data transfer. The data path loads new data parameters—that is, data expected, read/write data transfer, stream/block transfer, block size, byte count, card type, timeout registers—whenever a data transfer command is not in progress.

If the data\_expected bit is set in the Command register, the new command is a data transfer command and the data path starts one of the following:

- Transmit data if the read/write bit = 1
- Data receive if read/write bit = 0

## Data Transmit

The data transmit state machine, illustrated in following figure, starts data transmission two clocks after a response for the data write command is received; this occurs even if the command path detects a response error or response CRC error. If a response is not received from the card because of a response timeout, data is not transmitted. Depending upon the value of the transfer\_mode bit in the Command register, the data transmit state machine puts

data on the card data bus in a stream or in block(s).

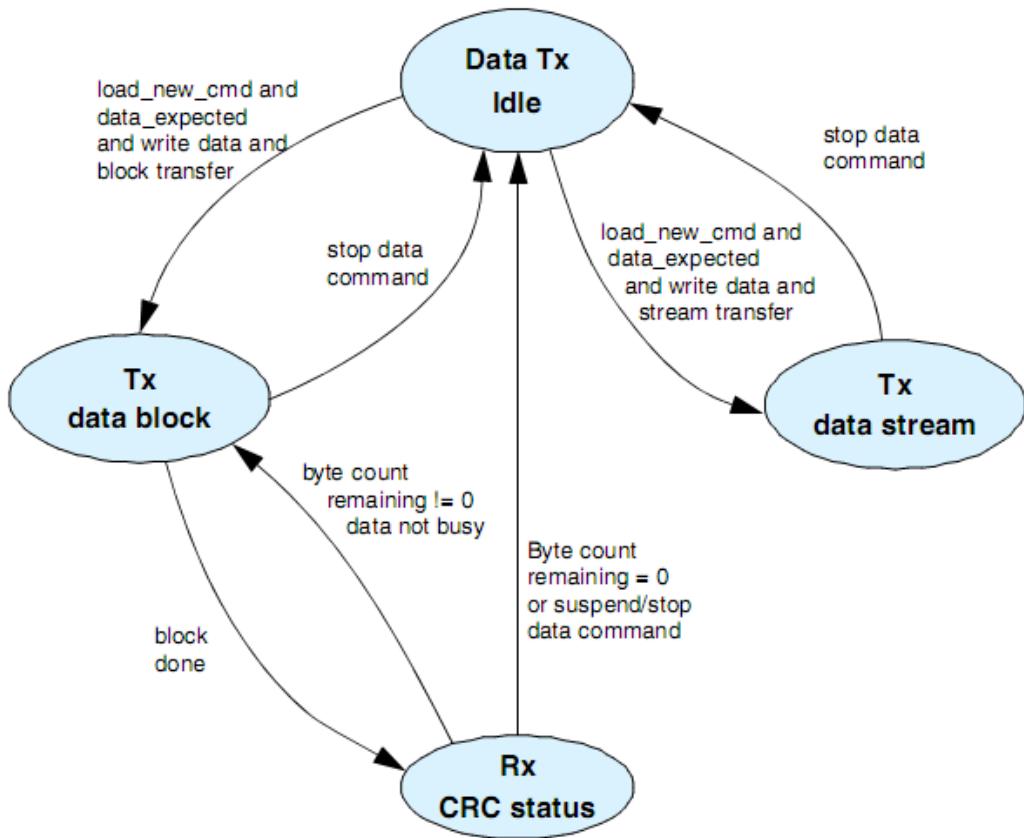


Fig. 14-4 Host Controller Data Transmit State Machine

### Stream Data Transmit

If the transfer\_mode bit in the Command register is set to 1, it is a stream-write data transfer. The data path pops the FIFO from the BIU and transmits in a stream to the card data bus. If the FIFO becomes empty, the card clock is stopped and restarted once data is available in the FIFO.

If the byte\_count register is programmed to 0, it is an open-ended stream-write data transfer. During this data transfer, the data path continuously transmits data in a stream until the host software issues a stop command. A stream data transfer is terminated when the end bit of the stop command and end bit of the data match over two clocks.

If the byte\_count register is programmed with a non-zero value and the send\_auto\_stop bit is set in the Command register, the stop command is internally generated and loaded in the command path when the end bit of the stop command occurs after the last byte of the stream write transfer matches.

This data transfer can also terminate if the host issues a stop command before all the data bytes are transferred to the card bus.

### Single Block Data

If the transfer\_mode bit in the Command register is set to 0 and the byte\_count register value is equal to the value of the block\_size register, a single-block write-data transfer occurs. The data transmit state machine sends data in a single block, where the number of bytes equals the block size, including the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the card\_num value in the

Command register – is set for a 1-bit, 4-bit, or 8-bit data transfer, the data is transmitted on 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and transmitted for 1, 4, or 8 data lines, respectively.

After a single data block is transmitted, the data transmit state machine receives the CRC status from the card and signals a data transfer to the BIU; this happens when the data-transfer-over bit is set in the RINTSTS register.

If a negative CRC status is received from the card, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register.

Additionally, if the start bit of the CRC status is not received by two clocks after the end of the data block, a CRC status start bit error is signaled to the BIU by setting the write-no-CRC bit in the RINTSTS register.

### **Multiple Block Data**

A multiple-block write-data transfer occurs if the transfer\_mode bit in the Command register is set to 0 and the value in the byte\_count register is not equal to the value of the block\_size register. The data transmit state machine sends data in blocks, where the number of bytes in a block equals the block size, including the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the card\_num value in the Command register – is set to 1-bit, 4-bit, or 8-bit data transfer, the data is transmitted on 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and transmitted on 1, 4, or 8 data lines, respectively.

After one data block is transmitted, the data transmit state machine receives the CRC status from the card. If the remaining byte\_count becomes 0, the data path signals to the BIU that the data transfer is done; this happens when the data-transfer-over bit is set in the RINTSTS register.

If the remaining data bytes are greater than 0, the data path state machine starts to transmit another data block.

If a negative CRC status is received from the card, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register, and continues further data transmission until all the bytes are transmitted.

Additionally, if the CRC status start bit is not received by two clocks after the end of a data block, a CRC status start bit error is signaled to the BIU by setting the write-no-CRC bit in the RINTSTS register; further data transfer is terminated.

If the send\_auto\_stop bit is set in the Command register, the stop command is internally generated during the transfer of the last data block, where no extra bytes are transferred to the card. The end bit of the stop command may not exactly match the end bit of the CRC status in the last data block.

If the block size is less than 4, 16, or 32 for card data widths of 1 bit, 4 bits, or 8 bits, respectively, the data transmit state machine terminates the data transfer when all the data is transferred, at which time the internally generated stop command is loaded in the command path.

If the byte\_count is 0 – the block size must be greater than 0 – it is an open-ended block transfer. The data transmit state machine for this type of data transfer continues the block-write data transfer until the host software issues a stop or abort command.

### **Data Receive**

The data-receive state machine, illustrated in following figure, receives data two clock cycles

after the end bit of a data read command, even if the command path detects a response error or response CRC error. If a response is not received from the card because a response timeout occurs, the BIU does not receive a signal that the data transfer is complete; this happens if the command sent by the Host Controller is an illegal operation for the card, which keeps the card from starting a read data transfer.

If data is not received before the data timeout, the data path signals a data timeout to the BIU and an end to the data transfer done. Based on the value of the transfer\_mode bit in the Command register, the data-receive state machine gets data from the card data bus in a stream or block(s).

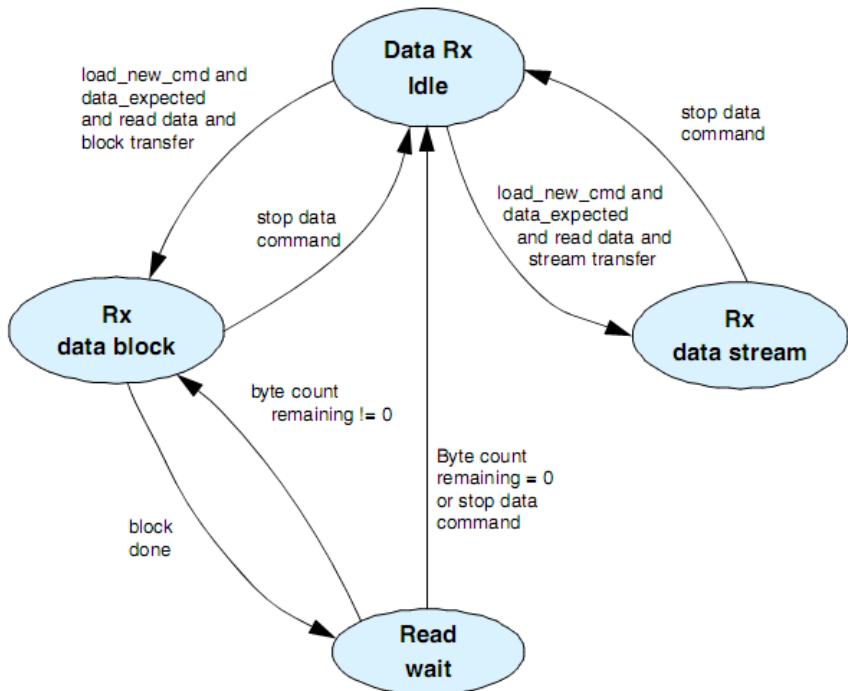


Fig. 14-5 Host Controller Data Receive State Machine

## Stream Data Read

A stream-read data transfer occurs if the transfer\_mode bit in the Command register equals 1, at which time the data path receives data from the card and pushes it to the FIFO. If the FIFO becomes full, the card clock stops and restarts once the FIFO is no longer full.

An open-ended stream-read data transfer occurs if the byte\_count register equals 0. During this type of data transfer, the data path continuously receives data in a stream until the host software issues a stop command. A stream data transfer terminates two clock cycles after the end bit of the stop command.

If the byte\_count register contains a non-zero value and the send\_auto\_stop bit is set in the Command register, a stop command is internally generated and loaded into the command path, where the end bit of the stop command occurs after the last byte of the stream data transfer is received. This data transfer can terminate if the host issues a stop or abort command before all the data bytes are received from the card.

## Single-Block Data Read

A single-block read-data transfer occurs if the transfer\_mode bit in the Command register is set to 0 and the value of the byte\_count register is equal to the value of the block\_size register. When a start bit is received before the data times out, data bytes equal to the block size and CRC16 are received and checked with the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the card\_num value in the Command register – is set to a 1-bit, 4-bit, or 8-bit data transfer, data is received from 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and checked for 1, 4, or 8 data lines, respectively. If there is a CRC16 mismatch, the data path signals a data CRC error to the BIU. If the received end bit is not 1, the BIU receives an end-bit error.

### Multiple-Block Data Read

If the transfer\_mode bit in the Command register is set to 0 and the value of the byte\_count register is not equal to the value of the block\_size register, it is a multiple-block read-data transfer. The data-receive state machine receives data in blocks, where the number of bytes in a block is equal to the block size, including the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the card\_num value in the Command register – is set to a 1-bit, 4-bit, or 8-bit data transfer, data is received from 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and checked for 1, 4, or 8 data lines, respectively.

After a data block is received, if the remaining byte\_count becomes 0, the data path signals a data transfer to the BIU.

If the remaining data bytes are greater than 0, the data path state machine causes another data block to be received. If CRC16 of a received data block does not match the internally-generated CRC16, a data CRC error to the BIU and data reception continue further data transmission until all bytes are transmitted.

Additionally, if the end of a received data block is not 1, data on the data path signals terminate the bit error to the CIU and the data-receive state machine terminates data reception, waits for data timeout, and signals to the BIU that the data transfer is complete.

If the send\_auto\_stop bit is set in the Command register, the stop command is internally generated when the last data block is transferred, where no extra bytes are transferred from the card; the end bit of the stop command may not exactly match the end bit of the last data block.

If the requested block size for data transfers to cards is less than 4, 16, or 32 bytes for 1-bit, 4-bit, or 8-bit data transfer modes, respectively, the data-transmit state machine terminates the data transfer when all data is transferred, at which point the internally-generated stop command is loaded in the command path. Data received from the card after that are then ignored by the data path.

If the byte\_count is 0—the block size must be greater than 0—it is an open-ended block transfer. For this type of data transfer, the data-receive state machine continues the block-read data transfer until the host software issues a stop or abort command.

### Auto-Stop

The Host Controller internally generates a stop command and is loaded in the command path when the send\_auto\_stop bit is set in the Command register.

The software should set the send\_auto\_stop bit according to details listed in following table.

Table 14-2 Auto-Stop Generation

Card type	Transfer type	Byte Count	send_auto_stop bit set	Comments
MMC	Stream read	0	No	Open-ended stream
MMC	Stream read	>0	Yes	Auto-stop after all bytes transfer
MMC	Stream write	0	No	Open-ended stream

MMC	Stream write	>0	Yes	Auto-stop after all bytes transfer
MMC	Single-block read	>0	No	Byte count =0 is illegal
MMC	Single-block write	>0	No	Byte count =0 is illegal
MMC	Multiple-block read	0	No	Open-ended multiple block
MMC	Multiple-block read	>0	Yes <sup>①</sup>	Pre-defined multiple block
MMC	Multiple-block write	0	No	Open-ended multiple block
MMC	Multiple-block write	>0	Yes <sup>①</sup>	Pre-defined multiple block
SDMEM	Single-block read	>0	No	Byte count =0 is illegal
SDMEM	Single-block write	>0	No	Byte count =0 illegal
SDMEM	Multiple-block read	0	No	Open-ended multiple block
SDMEM	Multiple-block read	>0	Yes	Auto-stop after all bytes transfer
SDMEM	Multiple-block write	0	No	Open-ended multiple block
SDMEM	Multiple-block write	>0	Yes	Auto-stop after all bytes transfer
SDIO	Single-block read	>0	No	Byte count =0 is illegal
SDIO	Single-block write	>0	No	Byte count =0 illegal
SDIO	Multiple-block read	0	No	Open-ended multiple block
SDIO	Multiple-block read	>0	No	Pre-defined multiple block
SDIO	Multiple-block write	0	No	Open-ended multiple block
SDIO	Multiple-block write	>0	No	Pre-defined multiple block

<sup>①</sup>:The condition under which the transfer mode is set to block transfer and byte\_count is equal to block size is treated as a single-block data transfer command for both MMC and SD cards. If byte\_count = n\*block\_size (n = 2, 3, ...), the condition is treated as a predefined multiple-block data transfer command. In the case of an MMC card, the host software can perform a predefined data transfer in two ways: 1) Issue the CMD23 command before issuing CMD18/CMD25 commands to the card – in this case, issue MD18/CMD25 commands without setting the send\_auto\_stop bit. 2) Issue CMD18/CMD25 commands without issuing CMD23 command to the card, with the send\_auto\_stop bit set. In this case, the multiple-block data transfer is terminated by an internally-generated auto-stop command after the programmed byte count.

The following list conditions for the auto-stop command.

- Stream read for MMC card with byte count greater than 0 – The Host Controller generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent out when the last byte of data is read from the card and no extra data byte is received. If the byte count is less than 6 (48 bits), a few extra data bytes are received from the card before the end bit of the stop command is sent.
- Stream write for MMC card with byte count greater than 0 - The Host Controller generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent when the last byte of data is transmitted on the card bus and no extra data byte is transmitted. If the byte count is less than 6 (48 bits), the data path transmits the data last in order to meet the above condition.
- Multiple-block read memory for SD card with byte count greater than 0 – If the block size is less than 4 (single-bit data bus), 16 (4-bit data bus), or 32 (8-bit data bus), the auto-stop command is loaded in the command path after all the bytes are read. Otherwise, the top command is loaded in the command path so that the end bit of the stop command is sent after the last data block is received.
- Multiple-block write memory for SD card with byte count greater than 0 – If the block size is less than 3 (single-bit data bus), 12 (4-bit data bus), or 24 (8-bit data bus), the auto-stop command is loaded in the command path after all data blocks are transmitted. Otherwise, the stop command is loaded in the command path so that the end bit of the stop command is sent after the end bit of the CRC status is received.
- Precaution for host software during auto-stop – Whenever an auto-stop command is issued, the host software should not issue a new command to the SD/MMC device until the auto-stop is sent by the Host Controller and the data transfer is complete. If the host issues a new command during a data transfer with the auto-stop in progress, an auto-stop command may be sent after the new command is sent and its response is received; this can delay sending the stop command, which transfers extra data bytes. For a stream write, extra data bytes are erroneous data that can corrupt the card data. If the host wants to terminate the data transfer before the data

transfer is complete, it can issue a stop or abort command, in which case the Host Controller does not generate an auto-stop command.

### 3. Non-Data Transfer Commands that Use Data Path

Some non-data transfer commands (non-read/write commands) also use the data path. Following table lists the commands and register programming requirements for them.

Table 14-3 Non-data Transfer Commands and Requirements

	CMD27	CMD30	CMD42	ACMD13	ACMD22	ACMD51
<b>Command register programming</b>						
cmd_index	6'h1B	6'h1E	6'h2A	6'h0D	6'h16	6'h33
response_expect	1	1	1	1	1	1
rResponse_length	0	0	0	0	0	0
check_response_crc	1	1	1	1	1	1
data_expected	1	1	1	1	1	1
read/write	1	0	1	0	0	0
transfer_mode	0	0	0	0	0	0
send_auto_stop	0	0	0	0	0	0
wait_prevdata_complete	0	0	0	0	0	0
stop_abort_cmd	0	0	0	0	0	0
<b>Command Argument register programming</b>						
	stuff bits	32-bit writeprotect dataaddress	stuff bits	stuff bits	stuff bits	stuff bits
<b>Block Size register programming</b>						
	16	4	Num_bytes <sup>①</sup>	64	4	8
<b>Byte Count register programming</b>						
	16	4	Num_bytes <sup>①</sup>	64	4	8

①: Num\_bytes = No. of bytes specified as per the lock card data structure

(Refer to the SD specification and the MMC specification)

### 4. SDIO Interrupt Control

Interrupts for SD cards are reported to the BIU by asserting an interrupt signal for two clock cycles. SDIO cards signal an interrupt by asserting cdata\_in low during the interrupt period; an interrupt period for the selected card is determined by the interrupt control state machine. An interrupt period is always valid for non-active or non-selected cards, and 1-bit data mode for the selected card. An interrupt period for a wide-bus active or selected card is valid for the following conditions:

- Card is idle
- Non-data transfer command in progress
- Third clock after end bit of data block between two data blocks
- From two clocks after end bit of last data until end bit of next data transfer command

Bear in mind that, in the following situations, the controller does not sample the SDIO interrupt of the selected card when the card data width is 4 bits. Since the SDIO interrupt is level-triggered, it is sampled in a further interrupt period and the host does not lose any SDIO

interrupt from the card.

- Read/Write Resume – The CIU treats the resume command as a normal data transfer command. SDIO interrupts during the resume command are handled similarly to other data commands. According to the SDIO specification, for the normal data command the interrupt period ends after the command end bit of the data command; for the resume command, it ends after the response end bit. In the case of the resume command, the Controller stops the interrupt sampling period after the resume command end bit, instead of stopping after the response end bit of the resume command.
- Suspend during read transfer – If the read data transfer is suspended by the host, the host sets the abort\_read\_data bit in the controller to reset the data state machine. In the CIU, the SDIO interrupts are handled such that the interrupt sampling starts after the abort\_read\_data bit is set by the host. In this case the controller does not sample SDIO interrupts between the period from response of the suspend command to setting the abort\_read\_data bit, and starts sampling after setting the abort\_read\_data bit.

## 5. Clock Control

The clock control block provides different clock frequencies required for SD/MMC cards. The cclk\_in signal is the source clock ( $cclk\_in \geq$  card max operating frequency) for clock divider of the clock control block. This source clock (cclk\_in) is used to generate different card clock frequencies (cclk\_out). The card clock can have different clock frequencies, since the card can be a low-speed card or a full-speed card. The Host Controller provides one clock signal (cclk\_out).

The clock frequency of a card depends on the following clock control registers:

- Clock Divider register – Internal clock dividers are used to generate different clock frequencies required for card. The division factor for each clock divider can be programmed by writing to the Clock Divider register. The clock divider is an 8-bit value that provides a clock division factor from 1 to 510; a value of 0 represents a clock-divider bypass, a value of 1 represents a divide by 2, a value of 2 represents a divide by 4, and so on.
- Clock Control register – cclk\_out can be enabled or disabled for each card under the following conditions:
  - clk\_enable – cclk\_out for a card is enabled if the clk\_enable bit for a card in the Clock Control register is programmed (set to 1) or disabled (set to 0).
  - Low-power mode – Low-power mode of a card can be enabled by setting the low-power mode bit of the Clock Control register to 1. If low-power mode is enabled to save card power, the cclk\_out is disabled when the card is idle for at least 8 card clock cycles. It is enabled when a new command is loaded and the command path goes to a non-idle state.

Additionally, cclk\_out is disabled when an internal FIFO is full – card read (no more data can be received from card) – or when the FIFO is empty – card write (no data is available for transmission). This helps to avoid FIFO overrun and underrun conditions. It is used by the command and data path to qualify cclk\_in for driving outputs and sampling inputs at the programmed clock frequency for the selected card, according to the Clock Divider and Clock Source register values.

Under the following conditions, the card clock is stopped or disabled, along with the active clk\_en, for the selected card:

- Clock can be disabled by writing to Clock Enable register (clk\_en bit = 1).
- If low-power mode is selected and card is idle, or not selected for 8 clocks.
- FIFO is full and data path cannot accept more data from the card and data transfer is incomplete –to avoid FIFO overrun.
- FIFO is empty and data path cannot transmit more data to the card and data transfer is incomplete – to avoid FIFO underrun.

## 6. Error Detection

- Response

- Response timeout – Response expected with response start bit is not received within programmed number of clocks in timeout register.
- Response CRC error – Response is expected and check response CRC requested; response CRC7 does not match with the internally-generated CRC7.
- Response error – Response transmission bit is not 0, command index does not match with the command index of the send command, or response end bit is not 1.
- Data transmit
  - No CRC status – During a write data transfer, if the CRC status start bit is not received two clocks after the end bit of the data block is sent out, the data path does the following:
    - ◆ Signals no CRC status error to the BIU
    - ◆ Terminates further data transfer
    - ◆ Signals data transfer done to the BIU
  - Negative CRC – If the CRC status received after the write data block is negative (that is, not 010), a data CRC error is signaled to the BIU and further data transfer is continued.
  - Data starvation due to empty FIFO – If the FIFO becomes empty during a write data transmission, or if the card clock is stopped and the FIFO remains empty for data timeout clocks, then a data-starvation error is signaled to the BIU and the data path continues to wait for data in the FIFO.
- Data receive
  - Data timeout – During a read-data transfer, if the data start bit is not received before the number of clocks that were programmed in the timeout register, the data path does the following:
    - ◆ Signals data-timeout error to the BIU
    - ◆ Terminates further data transfer
    - ◆ Signals data transfer done to BIU
  - Data start bit error – During a 4-bit or 8-bit read-data transfer, if the all-bit data line does not have a start bit, the data path signals a data start bit error to the BIU and waits for a data timeout, after which it signals that the data transfer is done.
  - Data CRC error – During a read-data-block transfer, if the CRC16 received does not match with the internally generated CRC16, the data path signals a data CRC error to the BIU and continues further data transfer.
  - Data end-bit error – During a read-data transfer, if the end bit of the received data is not 1, the data path signals an end-bit error to the BIU, terminates further data transfer, and signals to the BIU that the data transfer is done.
  - Data starvation due to FIFO full – During a read data transmission and when the FIFO becomes full, the card clock is stopped. If the FIFO remains full for data timeout clocks, a data starvation error is signaled to the BIU (Data Starvation by Host Timeout bit is set in RINTSTS Register) and the data path continues to wait for the FIFO to start to empty.

### 14.3.3 Internal Direct Memory Access Controller (IDMAC)

The Internal Direct Memory Access Controller (IDMAC) has a Control and Status Register (CSR) and a single Transmit/Receive engine, which transfers data from host memory to the device port and vice versa. The controller utilizes a descriptor to efficiently move data from source to destination with minimal Host CPU intervention. You can program the controller to interrupt the Host CPU in situations such as data Transmit and Receive transfer completion from the card, as well as other normal or error conditions.

The IDMAC and the Host driver communicate through a single data structure. CSR addresses 0x80 to 0x98 are reserved for host programming.

The IDMAC transfers the data received from the card to the Data Buffer in the Host memory, and it transfers Transmit data from the Data Buffer in the Host memory to the FIFO.

Descriptors that reside in the Host memory act as pointers to these buffers.

A data buffer resides in physical memory space of the Host and consists of complete data or partial data. Buffers contain only data, while buffer status is maintained in the descriptor. Data chaining refers to data that spans multiple data buffers. However, a single descriptor cannot span multiple data.

A single descriptor is used for both reception and transmission. The base address of the list is written into Descriptor List Base Address Register (DBADDR @0x88). A descriptor list is forward linked. The Last Descriptor can point back to the first entry in order to create a ring structure. The descriptor list resides in the physical memory address space of the Host. Each

descriptor can point to a maximum of two data buffers.

## 1. IDMAC CSR Access

When an IDMAC is introduced, an additional CSR space resides in the IDMAC that controls the IDMAC functionality. The host accesses the new CSR space in addition to the existing control register set in the BIU. The IDMAC CSR primarily contains descriptor information. For a write operation to the CSR, the respective CSR logic of the IDMAC and BIU decodes the address before accepting. For a read operation from the CSR, the appropriate CSR read path is enabled.

You can enable or disable the IDMAC operation by programming bit[25] in the CTRL register of the BIU. This allows the data transfer by accessing the slave interface on the AMBA bus if the IDMAC is present but disabled. When IDMAC is enabled, the FIFO cannot be accessed through the slave interface.

## 2. Descriptors

- Descriptor structures

The IDMAC uses these types of descriptor structures:

- Dual-Buffer Structure – The distance between two descriptors is determined by the Skip Length value programmed in the Descriptor Skip Length (DSL) field of the Bus Mode Register (BMOD @0x80).

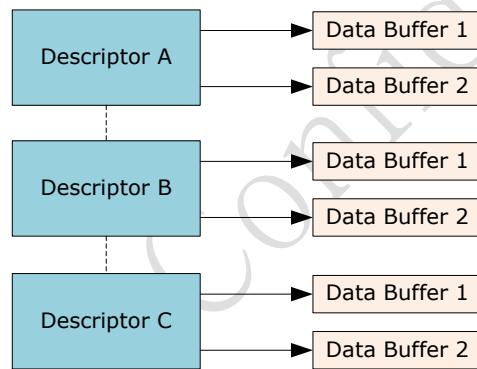


Fig. 14-6 Dual-Buffer Descriptor Structure

- Chain Structure – Each descriptor points to a unique buffer and the next descriptor.

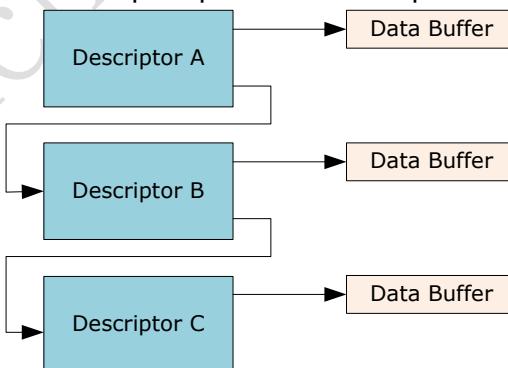


Fig. 14-7 Chain Descriptor Structure

- Descriptor formats

Following figure illustrates the internal formats of a descriptor. The descriptor addresses must be aligned to the bus width used for 32-bit AHB data buses. Each descriptor contains 16 bytes of control and status information. DES0 is a notation used to denote the [31:0] bits, DES1 to denote [63:32] bits, DES2 to denote [95:64] bits, DES3 to denote [127:96] bits.

Descriptor format  
for 32-bit bus width

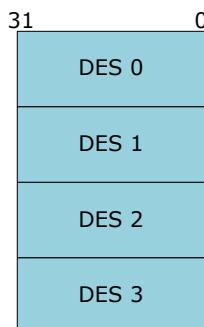


Fig. 14-8 Descriptor Formats for 32-bit AHB Address Bus Width

- The DES0 element in the IDMAC contains control and status information.

Table 14-4 Bits in IDMAC DES0 Element

Bits	Name	Description
31	OWN	When set, this bit indicates that the descriptor is owned by the IDMAC. When this bit is reset, it indicates that the descriptor is owned by the Host. The IDMAC clears this bit when it completes the data transfer.
30	Card Error Summary (CES)	These error bits indicate the status of the transaction to or from the card. These bits are also present in RINTSTS Indicates the logical OR of the following bits: <ul style="list-style-type: none"> <li>➤ EBE: End Bit Error</li> <li>➤ RTO: Response Time out</li> <li>➤ RCRC: Response CRC</li> <li>➤ SBE: Start Bit Error</li> <li>➤ DRTO: Data Read Timeout</li> <li>➤ DCRC: Data CRC for Receive</li> <li>➤ RE: Response Error</li> </ul>
29:6	Reserved	-
5	End of Ring (ER)	When set, this bit indicates that the descriptor list reached its final descriptor. The IDMAC returns to the base address of the list, creating a Descriptor Ring. This is meaningful for only a dual-buffer descriptor structure.
4	Second Address Chained (CH)	When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is set, BS2 (DES1[25:13]) should be all zeros.
3	First Descriptor (FS)	When set, this bit indicates that this descriptor contains the first buffer of the data. If the size of the first buffer is 0, next Descriptor contains the beginning of the data.

2	Last Descriptor (LD)	This bit is associated with the last block of a DMA transfer. When set, the bit indicates that the buffers pointed to by this descriptor are the last buffers of the data. After this descriptor is completed, the remaining byte count is 0. In other words, after the descriptor with the LD bit set is completed, the remaining byte count should be 0.
1	Disable Interrupt on Completion (DIC)	When set, this bit will prevent the setting of the TI/RI bit of the IDMAC Status Register (IDSTS) for the data that ends in the buffer pointed to by this descriptor.
0	Reserved	-

- The DES1 element contains the buffer size.

Table 14-5 Bits in IDMAC DES1 Element

Bits	Name	Description
31:26	Reserved	-
25:13	Buffer 2 Size (BS2)	These bits indicate the second data buffer byte size. The buffer size must be a multiple of 2, 4, or 8, depending upon the bus widths—16, 32, and 64, respectively. In the case where the buffer size is not a multiple of 2, 4, or 8, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and proceeds to the next buffer in case of a dual-buffer structure. This field is not valid for chain structure; that is, if DES0[4] is set.
12:0	Buffer 1 Size (BS1)	Indicates the data buffer byte size, which must be a multiple of 2, 4, or 8 bytes, depending upon the bus widths—16, 32, and 64, respectively. In the case where the buffer size is not a multiple of 2, 4, or 8, the resulting behavior is undefined. This field should not be zero. Note: If there is only one buffer to be programmed, you need to use only the Buffer 1, and not Buffer 2.

- The DES2 element contains the address pointer to the data buffer.

Table 14-6 Bits in IDMAC DES2 Element

Bits	Name	Description
31:26	Reserved	
25:13	Buffer 2 Size (BS2)	These bits indicate the second data buffer byte size. The buffer size must be a multiple of 2, 4, or 8, depending upon the bus widths—16, 32, and 64, respectively. In the case where the buffer size is not a multiple of 2, 4, or 8, the resulting behavior is

		undefined. If this field is 0, the DMA ignores this buffer and proceeds to the next buffer in case of a dual-buffer structure. This field is not valid for chain structure; that is, if DES0[4] is set.
12:0	Buffer 1 Size (BS1)	Indicates the data buffer byte size, which must be a multiple of 2, 4, or 8 bytes, depending upon the bus widths—16, 32, and 64, respectively. In the case where the buffer size is not a multiple of 2, 4, or 8, the resulting behavior is undefined. This field should not be zero. Note: If there is only one buffer to be programmed, you need to use only the Buffer 1, and not Buffer 2.

- The DES3 element contains the address pointer to the next descriptor if the present descriptor is not the last descriptor in a chained descriptor structure or the second buffer address for a dual-buffer structure.

Table 14-7 Bits in IDMAC DES3 Element

Bits	Name	Description
31:0	Buffer Address Pointer 2/ Next Descriptor Address (BAP2)	These bits indicate the physical address of the second buffer when the dual-buffer structure is used. If the Second Address Chained (DES0[4]) bit is set, then this address contains the pointer to the physical memory where the Next Descriptor is present. If this is not the last descriptor, then the Next Descriptor address pointer must be bus-width aligned.

### 3. Initialization

IDMAC initialization occurs as follows:

- 1) Write to IDMAC Bus Mode Register—BMOD to set Host bus access parameters.
  - 2) Write to IDMAC Interrupt Enable Register—IDINTEN to mask unnecessary interrupt causes.
  - 3) The software driver creates either the Transmit or the Receive descriptor list. Then it writes to IDMAC Descriptor List Base Address Register (DBADDR), providing the IDMAC with the starting address of the list.
  - 4) The IDMAC engine attempts to acquire descriptors from the descriptor lists.
- Host Bus Burst Access

The IDMAC attempts to execute fixed-length burst transfers on the AHB Master interface if configured using the FB bit of the IDMAC Bus Mode register. The maximum burst length is indicated and limited by the PBL field. The descriptors are always accessed in the maximum possible burst-size for the 16-bytes to be read— 16\*8/bus-width.

The IDMAC initiates a data transfer only when sufficient space to accommodate the configured burst is available in the FIFO or the number of bytes to the end of data, when less than the configured burst-length.

The IDMAC indicates the start address and the number of transfers required to the AHB Master Interface. When the AHB Interface is configured for fixed-length bursts, then it transfers data

using the best combination of INCR4/8/16 and SINGLE transactions. Otherwise, in no fixed-length bursts, it transfers data using INCR (undefined length) and SINGLE transactions.

- Host Data Buffer Alignment

The Transmit and Receive data buffers in host memory must be aligned, depending on the data width.

- Buffer Size Calculations

The driver knows the amount of data to transmit or receive. For transmitting to the card, the IDMAC transfers the exact number of bytes to the FIFO, indicated by the buffer size field of DES1.

If a descriptor is not marked as last-LS bit of DES0-then the corresponding buffer(s) of the descriptor are full, and the amount of valid data in a buffer is accurately indicated by its buffer size field. If a descriptor is marked as last, then the buffer cannot be full, as indicated by the buffer size in DES1. The driver is aware of the number of locations that are valid in this case.

- Transmission

IDMAC transmission occurs as follows:

- 1) The Host sets up the elements (DES0-DES3) for transmission and sets the OWN bit (DES0[31]). The Host also prepares the data buffer.
- 2) The Host programs the write data command in the CMD register in BIU.
- 3) The Host will also program the required transmit threshold level (TX\_WMark field in FIFOTH register).
- 4) The IDMAC determines that a write data transfer needs to be done as a consequence of step 2.
- 5) The IDMAC engine fetches the descriptor and checks the OWN bit. If the OWN bit is not set, it means that the host owns the descriptor. In this case the IDMAC enters suspend state and asserts the Descriptor Unable interrupt in the IDSTS register. In such a case, the host needs to release the IDMAC by writing any value to the poll demand register.
- 6) It will then wait for Command Done (CD) bit and no errors from BIU which indicates that a transfer can be done.
- 7) The IDMAC engine will now wait for a DMA interface request from BIU. This request will be generated based on the programmed transmit threshold value. For the last bytes of data which can't be accessed using a burst, SINGLE transfers are performed on AHB Master Interface.
- 8) The IDMAC fetches the Transmit data from the data buffer in the Host memory and transfers to the FIFO for transmission to card.
- 9) When data spans across multiple descriptors, the IDMAC will fetch the next descriptor and continue with its operation with the next descriptor. The Last Descriptor bit in the descriptor indicates whether the data spans multiple descriptors or not.
- 10) When data transmission is complete, status information is updated in IDSTS register by setting Transmit Interrupt, if enabled. Also, the OWN bit is cleared by the IDMAC by performing a write transaction to DES0.

- Reception

IDMAC reception occurs as follows:

- 1) The Host sets up the element (DES0-DES3) for reception, sets the OWN (DES0[31]).
- 2) The Host programs the read data command in the CMD register in BIU.
- 3) The Host will program the required receive threshold level (RX\_WMark field in FIFOTH register).
- 4) The IDMAC determines that a read data transfer needs to be done as a consequence of step 2.
- 5) The IDMAC engine fetches the descriptor and checks the OWN bit. If the OWN bit is not set, it means that the host owns the descriptor. In this case the DMA enters suspend state and asserts the Descriptor Unable interrupt in the IDSTS register. In such a case, the host needs to release the IDMAC by writing any value to the poll demand

register.

- 6) It will then wait for Command Done (CD) bit and no errors from BIU which indicates that a transfer can be done.
- 7) The IDMAC engine will now wait for a DMA interface request from BIU. This request will be generated based on the programmed receive threshold value. For the last bytes of data which can't be accessed using a burst, SINGLE transfers are performed on AHB.
- 8) The IDMAC fetches the data from the FIFO and transfer to Host memory.
- 9) When data spans across multiple descriptors, the IDMAC will fetch the next descriptor and continue with its operation with the next descriptor. The Last Descriptor bit in the descriptor indicates whether the data spans multiple descriptors or not.
- 10) When data reception is complete, status information is updated in IDSTS register by setting Receive Interrupt, if enabled. Also, the OWN bit is cleared by the IDMAC by performing a write transaction to DES0.

- **Interrupts**

Interrupts can be generated as a result of various events. IDSTS register contains all the bits that might cause an interrupt. IDINTEN register contains an Enable bit for each of the events that can cause an interrupt.

There are two groups of summary interrupts-Normal and Abnormal-as outlined in IDSTS register. Interrupts are cleared by writing a 1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary bits are cleared, the interrupt signal dmac\_intr\_o is de-asserted.

Interrupts are not queued and if the interrupt event occurs before the driver has responded to it, no additional interrupts are generated. For example, Receive Interrupt—IDSTS[1] indicates that one or more data was transferred to the Host buffer.

An interrupt is generated only once for simultaneous, multiple events. The driver must scan IDSTS register for the interrupt cause.

## 14.4 Register Description

### 14.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
SDMMC_CTRL	0x0000	W	0x01000000	Control register
SDMMC_PWREN	0x0004	W	0x00000000	Power-enable register
SDMMC_CLKDIV	0x0008	W	0x00000000	Clock-divider register
SDMMC_CLKSRC	0x000c	W	0x00000000	SD Clock Source Register
SDMMC_CLKENA	0x0010	W	0x00000000	Clock-enable register
SDMMC_TMOUT	0x0014	W	0xfffffff40	Time-out register
SDMMC_CTYPE	0x0018	W	0x00000000	Card-type register
SDMMC_BLKSIZ	0x001c	W	0x00000200	Block-size register
SDMMC_BYTCNT	0x0020	W	0x00000200	Byte-count register
SDMMC_INTMASK	0x0024	W	0x00000000	Interrupt-mask register
SDMMC_CMDARG	0x0028	W	0x00000000	Command-argument register
SDMMC_CMD	0x002c	W	0x00000000	Command register
SDMMC_RESP0	0x0030	W	0x00000000	Response-0 register
SDMMC_RESP1	0x0034	W	0x00000000	Response-1 register
SDMMC_RESP2	0x0038	W	0x00000000	Response-2 register

Name	Offset	Size	Reset Value	Description
SDMMC_RESP3	0x003c	W	0x00000000	Response-3 register
SDMMC_MINTSTS	0x0040	W	0x00000000	Masked interrupt-status register
SDMMC_RINTSTS	0x0044	W	0x00000000	Raw interrupt-status register
SDMMC_STATUS	0x0048	W	0x00000406	Status register
SDMMC_FIFOTH	0x004c	W	0x00000000	FIFO threshold register
SDMMC_CDETECT	0x0050	W	0x00000000	Card-detect register
SDMMC_WRTPRT	0x0054	W	0x00000000	Write-protect register
SDMMC_TCBCNT	0x005c	W	0x00000000	Transferred CIU card byte count
SDMMC_TBBCNT	0x0060	W	0x00000000	Transferred host/DMA to/from BIU-FIFO byte count
SDMMC_DEBNCE	0x0064	W	0x00ffffff	Card detect debounce register
SDMMC_USRID	0x0068	W	0x07967797	User ID register
SDMMC_VERID	0x006c	W	0x5342270a	Synopsys version ID register
SDMMC_HCON	0x0070	W	0x00000000	Hardware Configuration Register
SDMMC_UHS_REG	0x0074	W	0x00000000	UHS-1 register
SDMMC_RST_N	0x0078	W	0x00000001	Hardware reset register
SDMMC_BMOD	0x0080	W	0x00000000	Bus Mode Register
SDMMC_PLDMND	0x0084	W	0x00000000	Poll Demand Register
SDMMC_DBADDR	0x0088	W	0x00000000	Descriptor List Base Address Register
SDMMC_IDSTS	0x008c	W	0x00000000	Internal DMAC Status Register
SDMMC_IDINTEN	0x0090	W	0x00000000	Internal DMAC Interrupt Enable Register
SDMMC_DSCADDR	0x0094	W	0x00000000	Current Host Descriptor Address Register
SDMMC_BUFADDR	0x0098	W	0x00000000	Current Buffer Descriptor Address Register
SDMMC_CARDTHRCTL	0x0100	W	0x00000000	Card read threshold enable
SDMMC_BACK_END_POWER	0x0104	W	0x00000000	Back-end power
SDMMC_UHS_REG_EXT	0x0108	W	0x00000000	UHS Register
SDMMC_EMMC_DDR_REG	0x010c	W	0x00000000	eMMC 4.5 DDR START Bit Detection Control Register
SDMMC_ENABLE_SHIFT	0x0110	W	0x00000000	Enable Phase Shift Register
SDMMC_FIFO_BASE	0x0200	W	0x00000000	FIFO Base Address

Notes: **Size** : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

#### 14.4.2 Detail Register Description

##### SDMMC\_CTRL

Address: Operational Base + offset (0x0000)

Control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RO	0x0	reserved
25	RW	0x0	use_internal_dmac Present only for the Internal DMAC configuration; else, it is reserved. 0: The host performs data transfers through the slave interface 1: Internal DMAC used for data transfe
24	RW	0x1	enable_OD_pullup External open-drain pullup: 0: Disable 1: Enable Inverted value of this bit is output to ccmd_od_pullup_en_n port. When bit is set, command output always driven in open-drive mode; that is, DWC_mobile_storage drives either 0 or high impedance, and does not drive hard 1.
23:20	RW	0x0	Card_voltage_b Card regulator-B voltage setting; output to card_volt_b port. Optional feature; ports can be used as general-purpose outputs.
19:16	RW	0x0	Card_voltage_a Card regulator-A voltage setting; output to card_volt_a port. Optional feature; ports can be used as general-purpose outputs.
15:12	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x0	<p>ceata_device_interrupt_status</p> <p>0: Interrupts not enabled in CE-ATA device (nIEN = 1 in ATA control register)</p> <p>1: Interrupts are enabled in CE-ATA device (nIEN = 0 in ATA control register)</p> <p>Software should appropriately write to this bit after power-on reset or any other reset to CE-ATA device. After reset, usually CE-ATA device interrupt is disabled (nIEN = 1). If the host enables CE-ATA device interrupt, then software should set this bit.</p>
10	RW	0x0	<p>send_auto_stop_ccsd</p> <p>0: Clear bit if DWC_mobile_storage does not reset the bit.</p> <p>1: Send internally generated STOP after sending CCSD to CE-ATA device.</p> <p>NOTE: Always set send_auto_stop_ccsd and send_ccsd bits together send_auto_stop_ccsd should not be set independent of send_ccsd. When set, DWC_Mobile_Storage automatically sends internally-generated STOP command (CMD12) to CE-ATA device. After sending internally-generated STOP command, Auto Command Done (ACD) bit in RINTSTS is set and generates interrupt to host if Auto Command Done interrupt is not masked. After sending the CCSD, DWC_mobile_storage automatically clears send_auto_stop_ccsd bit.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	<p>send_ccsd            0: Clear bit if DWC_mobile_storage does not reset the bit.            1: Send Command Completion Signal Disable (CCSD) to CE-ATA device            When set, DWC_mobile_storage sends CCSD to CE-ATA device. Software sets this bit only if current command is expecting CCS (that is, RW_BLK) and interrupts are enabled in CE-ATA device. Once the CCSD pattern is sent to device, DWC_mobile_storage automatically clears send_ccsd bit. It also sets Command Done (CD) bit in RINTSTS register and generates interrupt to host if Command Done interrupt is not masked.</p> <p>NOTE: Once send_ccsd bit is set, it takes two card clock cycles to drive the CCSD on the CMD line. Due to this, during the boundary conditions it may happen that CCSD is sent to the CE-ATA device, even if the device signalled CCS</p>
8	RW	0x0	<p>abort_read_data            0: no change            1: after suspend command is issued during read-transfer, software polls card to find when suspend happened. Once suspend occurs, software sets bit to reset data state-machine, which is waiting for next block of data. Bit automatically clears once data state machine resets to idle.            Used in SDIO card suspend sequence.</p>
7	RW	0x0	<p>send_irq_response            0: no change            1: send auto IRQ response            Bit automatically clears once response is sent. To wait for MMC card interrupts, host issues CMD40, and SDMMC Controller waits for interrupt response from MMC card(s). In meantime, if host wants SDMMC Controller to exit waiting for interrupt state, it can set this bit, at which time SDMMC Controller command state-machine sends CMD40 response on bus and returns to idle state.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RW	0x0	read_wait 0: clear read wait 1: assert read wait For sending read-wait to SDIO cards
5	RW	0x0	dma_enable 0: disable DMA transfer mode 1: enable DMA transfer mode Even when DMA mode is enabled, host can still push/pop data into or from FIFO; this should not happen during the normal operation. If there is simultaneous FIFO access from host/DMA, the data coherency is lost. Also, there is no arbitration inside SDMMC Controller to prioritize simultaneous host/DMA access.
4	RW	0x0	int_enable Global interrupt enable/disable bit: 0: disable interrupts 1: enable interrupts The int port is 1 only when this bit is 1 and one or more unmasked interrupts are set.
3	RO	0x0	reserved
2	W1C	0x0	dma_reset 0: no change 1: reset internal DMA interface control logic To reset DMA interface, firmware should set bit to 1. This bit is auto-cleared after two AHB clocks.
1	W1C	0x0	fifo_reset 0: no change 1: reset to data FIFO To reset FIFO pointers To reset FIFO, firmware should set bit to 1. This bit is auto-cleared after completion of reset operation

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	W1C	0x0	<p>controller_reset 0: no change 1: reset SDMMC controller To reset controller, firmware should set bit to 1. This bit is auto-cleared after two AHB and two cclk_in clock cycles.</p> <p>This resets:            * BIU/CIU interface            * CIU and state machines            * abort_read_data, send_irq_response, and read_wait bits of Control register            * start_cmd bit of Command register            Does not affect any registers or DMA interface, or FIFO or host interrupts</p>

**SDMMC\_PWREN**

Address: Operational Base + offset (0x0004)

Power-enable register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	<p>power_enable Power on/off switch for the card. Once power is turned on, firmware should wait for regulator/switch ramp-up time before trying to initialize card.</p> <p>0: power off 1: power on Bit values output to card_power_en port.</p>

**SDMMC\_CLKDIV**

Address: Operational Base + offset (0x0008)

Clock-divider register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	<p>clk_divider3 Clock divider-3 value. For example, value of 0 means divide by <math>2^0 = 0</math> (no division, bypass), value of 1 means divide by <math>2^1 = 2</math>, value of "ff" means divide by <math>2^{255} = 510</math>, and so on In MMC-Ver3.3-only mode, bits not implemented because only one clock divider is supported. In our design, the divider is 0 or 1 supported.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23:16	RW	0x00	<p>clk_divider2 Clock divider-2 value. Clock division is <math>2^n</math>. For example, value of 0 means divide by <math>2^0 = 0</math> (no division, bypass), value of 1 means divide by <math>2^1 = 2</math>, value of "ff" means divide by <math>2^{255} = 510</math>, and so on</p> <p>In MMC-Ver3.3-only mode, bits not implemented because only one clock divider is supported.</p> <p>In our design, the divider is 0 or 1 supported.</p>
15:8	RW	0x00	<p>clk_divider1 Clock divider-1 value. Clock division is <math>2^n</math>. For example, value of 0 means divide by <math>2^0 = 0</math> (no division, bypass), value of 1 means divide by <math>2^1 = 2</math>, value of "ff" means divide by <math>2^{255} = 510</math>, and so on</p> <p>In MMC-Ver3.3-only mode, bits not implemented because only one clock divider is supported.</p> <p>In our design, the divider is 0 or 1 supported.</p>
7:0	RW	0x00	<p>clk_divider0 Clock divider-0 value. Clock division is <math>2^n</math>. For example, value of 0 means divide by <math>2^0 = 0</math> (no division, bypass), value of 1 means divide by <math>2^1 = 2</math>, value of "ff" means divide by <math>2^{255} = 510</math>, and so on.</p> <p>In our design, the divider is 0 or 1 supported.</p>

**SDMMC\_CLKSRC**

Address: Operational Base + offset (0x000c)

SD Clock Source Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RW	0x0	<p>clk_source Clock divider source for up to 16 SD cards supported. Each card has two bits assigned to it. For example, bits[1:0] assigned for card-0, which maps and internally routes clock divider[3:0] outputs to cclk_out[15:0] pins, depending on bit value.</p> <ul style="list-style-type: none"> <li>00 –Clock divider 0</li> <li>01 –Clock divider 1</li> <li>10 –Clock divider 2</li> <li>11 –Clock divider 3</li> </ul> <p>In our design, the cclk_out is always from clock divider 0, and this register is not implemented.</p>

**SDMMC\_CLKENA**

Address: Operational Base + offset (0x0010)

Clock-enable register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved
16	RW	0x0	<p>cclk_low_power Low-power control for SD card clock and MMC card clock supported. 0: non-low-power mode 1: low-power mode; stop clock when card in IDLE (should be normally set to only MMC and SD memory cards; for SDIO cards, if interrupts must be detected, clock should not be stopped).</p>
15:1	RO	0x0	reserved
0	RW	0x0	<p>cclk_enable Clock-enable control for SD card clock and MMC card clock supported. 0: clock disabled 1: clock enabled</p>

**SDMMC\_TMOUT**

Address: Operational Base + offset (0x0014)

Time-out register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RW	0xffffffff	<p>data_timeout Value for card Data Read Timeout; same value also used for Data Starvation by Host timeout. Value is in number of card output clocks cclk_out of selected card.</p> <p>Note: The software timer should be used if the timeout value is in the order of 100 ms. In this case, read data timeout interrupt needs to be disabled.</p>
7:0	RW	0x40	<p>response_timeout Response timeout value. Value is in number of card output clocks -cclk_out.</p>

**SDMMC\_CTYPE**

Address: Operational Base + offset (0x0018)

Card-type register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved
16	RW	0x0	<p>card_width_8 Indicates if card is 8-bit: 0: non 8-bit mode 1: 8-bit mode</p>
15:1	RO	0x0	reserved
0	RW	0x0	<p>card_width Indicates if card is 1-bit or 4-bit: 0: 1-bit mode 1: 4-bit mode</p>

**SDMMC\_BLKSIZ**

Address: Operational Base + offset (0x001c)

Block-size register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0200	block_size Block size

**SDMMC\_BYTCNT**

Address: Operational Base + offset (0x0020)

Byte-count register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000200	<p>byte_count Number of bytes to be transferred; should be integer multiple of Block Size for block transfers.</p> <p>For undefined number of byte transfers, byte count should be set to 0. When byte count is set to 0, it is responsibility of host to explicitly send stop/abort command to terminate data transfer.</p>

**SDMMC\_INTMASK**

Address: Operational Base + offset (0x0024)

Interrupt-mask register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	<p>sdio_int_mask Mask SDIO interrupts. When masked, SDIO interrupt detection for that card is disabled. A 0 masks an interrupt, and 1 enables an interrupt.</p>
23:17	RO	0x0	reserved
16	RW	0x0	<p>data_nobusy_int_mask 0: data no busy interrupt not masked 1: data no busy interrupt masked</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	<p>int_mask Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.</p> <p>bit 15: End-bit error (read)/Write no CRC (EBE) bit 14: Auto command done (ACD) bit 13: Start-bit error (SBE) bit 12: Hardware locked write error (HLE) bit 11: FIFO underrun/overrun error (FRUN) bit 10: Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9: Data read timeout (DRTO) bit 8: Response timeout (RTO) bit 7: Data CRC error (DCRC) bit 6: Response CRC error (RCRC) bit 5: Receive FIFO data request (RXDR) bit 4: Transmit FIFO data request (TXDR) bit 3: Data transfer over (DTO) bit 2: Command done (CD) bit 1: Response error (RE) bit 0: Card detect (CD)</p>

**SDMMC\_CMDARG**

Address: Operational Base + offset (0x0028)

Command-argument register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	cmd_arg Value indicates command argument to be passed to card.

**SDMMC\_CMD**

Address: Operational Base + offset (0x002c)

Command register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	<p>start_cmd Start command. Once command is taken by CIU, bit is cleared.</p> <p>When bit is set, host should not attempt to write to any command registers. If write is attempted, hardware lock error is set in raw interrupt register.</p> <p>Once command is sent and response is received from SD_MMC cards, Command Done bit is set in raw interrupt register.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
30	RO	0x0	reserved
29	RW	0x0	<p>use_hold_reg Use Hold Register 0: CMD and DATA sent to card bypassing HOLD Register 1: CMD and DATA sent to card through the HOLD Register</p> <p>Note:</p> <ul style="list-style-type: none"> <li>a. Set to 1'b1 for SDR12 and SDR25 (with non-zero phase-shifted cclk_in_drv); zero phase shift is not allowed in these modes.</li> <li>b. Set to 1'b0 for SDR50, SDR104, and DDR50 (with zero phase-shifted cclk_in_drv)</li> <li>c. Set to 1'b1 for SDR50, SDR104, and DDR50 (with non-zero phase-shifted cclk_in_drv)</li> </ul>
28	RW	0x0	<p>volt_switch Voltage switch bit. 0: no voltage switching 1: voltage switching enabled; must be set for CMD11 only</p>
27	RW	0x0	<p>boot_mode Boot Mode. 0: mandatory Boot operation 1: alternate Boot operation</p>
26	RW	0x0	<p>disable_boot Disable Boot. When software sets this bit along with start_cmd, CIU terminates the boot operation. Do NOT set disable_boot and enable_boot together.</p>
25	RW	0x0	<p>expect_boot_ack Expect Boot Acknowledge. When Software sets this bit along with enable_boot, CIU expects a boot acknowledge start pattern of 0-1-0 from the selected card.</p>
24	RW	0x0	<p>enable_boot Enable Boot—this bit should be set only for mandatory boot mode. When Software sets this bit along with start_cmd, CIU starts the boot sequence for the corresponding card by asserting the CMD line low. Do NOT set disable_boot and enable_boot together.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23	RW	0x0	<p>ccs_expected            0: Interrupts are not enabled in CE-ATA device (nIEN = 1 in ATA control register), or command does not expect CCS from device            1: Interrupts are enabled in CE-ATA device (nIEN = 0), and RW_BLK command expects command completion signal from CE-ATA device</p> <p>If the command expects Command Completion Signal (CCS) from the CE-ATA device, the software should set this control bit.</p> <p>DWC_mobile_storage sets Data Transfer Over (DTO) bit in RINTSTS register and generates interrupt to host if Data Transfer Over interrupt is not masked.</p>
22	RW	0x0	<p>read_ceata_device            0: Host is not performing read access (RW_REG or RW_BLK) towards CE-ATA device            1: Host is performing read access (RW_REG or RW_BLK) towards CE-ATA device</p> <p>Software should set this bit to indicate that CE-ATA device is being accessed for read transfer. This bit is used to disable read data timeout indication while performing CE-ATA read transfers.</p> <p>Maximum value of I/O transmission delay can be no less than 10 seconds.</p> <p>DWC_mobile_storage should not indicate read data timeout while waiting for data from CE-ATA device.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
21	RW	0x0	<p>update_clock_registers_only 0: normal command sequence 1: do not send commands, just update clock register value into card clock domain Following register values transferred into card clock domain: CLKDIV, CLRSRC, CLKENA. Changes card clocks (change frequency, truncate off or on, and set low-frequency mode); provided in order to change clock frequency or stop clock without having to send command to cards.</p> <p>During normal command sequence, when update_clock_registers_only = 0, following control registers are transferred from BIU to CIU: CMD, CMDARG, TMOUT, CTYPE, BLKSIZ, BYTCNT. CIU uses new register values for new command sequence to card. When bit is set, there are no Command Done interrupts because no command is sent to SD_MMC_CEATA cards.</p>
20:16	RW	0x00	<p>card_number Card number in use. Represents physical slot number of card being accessed. In MMC-Ver3.3-only mode, up to 30 cards are supported; in SD-only mode, up to 16 cards are supported. Registered version of this is reflected on dw_dma_card_num and ge_dma_card_num ports, which can be used to create separate DMA requests, if needed. In addition, in SD mode this is used to mux or demux signals from selected card because each card is interfaced to DWC_mobile_storage by separate bus.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RW	0x0	<p>send_initialization            0: do not send initialization sequence (80 clocks of 1) before sending this command            1: send initialization sequence before sending this command</p> <p>After power on, 80 clocks must be sent to card for initialization before sending any commands to card. Bit should be set while sending first command to card so that controller will initialize clocks before sending command to card. This bit should not be set for either of the boot modes (alternate or mandatory).</p>
14	RW	0x0	<p>stop_abort_cmd            0: neither stop nor abort command to stop current data transfer in progress. If abort is sent to function-number currently selected or not in data-transfer mode, then bit should be set to 0.            1: stop or abort command intended to stop current data transfer in progress.</p> <p>When open-ended or predefined data transfer is in progress, and host issues stop or abort command to stop data transfer, bit should be set so that command/data state-machines of CIU can return correctly to idle state. This is also applicable for Boot mode transfers. To Abort boot mode, this bit should be set along with CMD[26] = disable_boot.</p>
13	RW	0x0	<p>wait_prvdata_complete            0: send command at once, even if previous data transfer has not completed            1: wait for previous data transfer completion before sending command</p> <p>The wait_prvdata_complete = 0 option typically used to query status of card during data transfer or to stop current data transfer; card_number should be same as in previous command.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x0	<p>send_auto_stop 0: no stop command sent at end of data transfer 1: send stop command at end of data transfer When set, SDMMC Controller sends stop command to SD_MMC cards at end of data transfer.</p> <ul style="list-style-type: none"> <li>* when send_auto_stop bit should be set, since some data transfers do not need explicit stop commands</li> <li>* open-ended transfers that software should explicitly send to stop command</li> </ul> <p>Additionally, when "resume" is sent to resume –suspended memory access of SD-Combo card –bit should be set correctly if suspended data transfer needs send_auto_stop. Don't care if no data expected from card.</p>
11	RW	0x0	<p>transfer_mode 0: block data transfer command 1: stream data transfer command Don't care if no data expected.</p>
10	RW	0x0	<p>wr 0: read from card 1: write to card Don't care if no data expected from card.</p>
9	RW	0x0	<p>data_expected 0: no data transfer expected (read/write) 1: data transfer expected (read/write)</p>
8	RW	0x0	<p>check_response_crc 0: do not check response CRC 1: check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller</p>
7	RW	0x0	<p>response_length 0: short response expected from card 1: long response expected from card</p>
6	RW	0x0	<p>response_expect 0: no response expected from card 1: response expected from card</p>
5:0	RW	0x00	<p>cmd_index Command index</p>

**SDMMC\_RESP0**

Address: Operational Base + offset (0x0030)

Response-0 register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	response0 Bit[31:0] of response

### **SDMMC\_RESP1**

Address: Operational Base + offset (0x0034)

Response-1 register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	response Register represents bit[63:32] of long response. When CIU sends auto-stop command, then response is saved in register. Response for previous command sent by host is still preserved in Response 0 register. Additional auto-stop issued only for data transfer commands, and response type is always "short" for them.

### **SDMMC\_RESP2**

Address: Operational Base + offset (0x0038)

Response-2 register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	response2 Bit[95:64] of long response

### **SDMMC\_RESP3**

Address: Operational Base + offset (0x003c)

Response-3 register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	response3 Bit[127:96] of long response

### **SDMMC\_MINTSTS**

Address: Operational Base + offset (0x0040)

Masked interrupt-status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
24	RO	0x0	sdio_interrupt Interrupt from SDIO card; SDIO interrupt for card enabled only if corresponding sdio_int_mask bit is set in Interrupt mask register (mask bit 1 enables interrupt; 0 masks interrupt). 0: no SDIO interrupt from card 1: SDIO interrupt from card
23:17	RO	0x0	reserved
16	RW	0x0	data_nobusy_int_status Data no busy Interrupt Status
15:0	RO	0x0000	int_status Interrupt enabled only if corresponding bit in interrupt mask register is set. bit 15: End-bit error (read)/Write no CRC (EBE) bit 14: Auto command done (ACD) bit 13: Start-bit error (SBE) bit 12: Hardware locked write error (HLE) bit 11: FIFO underrun/overrun error (FRUN) bit 10: Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9: Data read timeout (DRTO) bit 8: Response timeout (RTO) bit 7: Data CRC error (DCRC) bit 6: Response CRC error (RCRC) bit 5: Receive FIFO data request (RXDR) bit 4: Transmit FIFO data request (TXDR) bit 3: Data transfer over (DTO) bit 2: Command done (CD) bit 1: Response error (RE) bit 0: Card detect (CD)

**SDMMC\_RINTSTS**

Address: Operational Base + offset (0x0044)

Raw interrupt-status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
24	RO	0x0	sdio_interrupt Interrupt from SDIO card; Writes to these bits clear them. Value of 1 clears bit and 0 leaves bit intact. 0: no SDIO interrupt from card 1: SDIO interrupt from card
23:17	RO	0x0	reserved
16	RW	0x0	data_nobusy_int_status Data no busy interrupt status
15:0	RO	0x0000	int_status Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. bit 15: End-bit error (read)/Write no CRC (EBE) bit 14: Auto command done (ACD) bit 13: Start-bit error (SBE) bit 12: Hardware locked write error (HLE) bit 11: FIFO underrun/overrun error (FRUN) bit 10: Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9: Data read timeout (DRTO) bit 8: Response timeout (RTO) bit 7: Data CRC error (DCRC) bit 6: Response CRC error (RCRC) bit 5: Receive FIFO data request (RXDR) bit 4: Transmit FIFO data request (TXDR) bit 3: Data transfer over (DTO) bit 2: Command done (CD) bit 1: Response error (RE) bit 0: Card detect (CD)

**SDMMC\_STATUS**

Address: Operational Base + offset (0x0048)

Status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	dma_req DMA request signal state
30	RO	0x0	dma_ack DMA acknowledge signal state
29:17	RO	0x0000	fifo_count Number of filled locations in FIFO

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16:11	RO	0x00	response_index Index of previous response, including any auto-stop sent by core
10	RO	0x1	data_state_mc_busy Data transmit or receive state-machine is busy
9	RO	0x0	data_busy Inverted version of raw selected card_data[0] 0: card data not busy 1: card data busy default value is 1 or 0 depending on cdata_in
8	RO	0x0	data_3_status Raw selected card_data[3]; checks whether card is present 0: card not present 1: card present default value is 1 or 0 depending on cdata_in

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:4	RO	0x0	<p>command_fsm_states Command FSM states: 0: idle 1: send init sequence 2: Tx cmd start bit 3: Tx cmd tx bit 4: Tx cmd index + arg 5: Tx cmd crc7 6: Tx cmd end bit 7: Rx resp start bit 8: Rx resp IRQ response 9: Rx resp tx bit 10: Rx resp cmd idx 11: Rx resp data 12: Rx resp crc7 13: Rx resp end bit 14: Cmd path wait NCC 15: Wait; CMD-to-response turnaround</p> <p>The command FSM state is represented using 19 bits.</p> <p>The STATUS Register[7:4] has 4 bits to represent the command FSM states. Using these 4 bits, only 16 states can be represented. Thus three states cannot be represented in the STATUS[7:4] register. The three states that are not represented in the STATUS Register[7:4] are:</p> <ul style="list-style-type: none"> <li>* Bit 16 –Wait for CCS</li> <li>* Bit 17 –Send CCSD</li> <li>* Bit 18 –Boot Mode</li> </ul> <p>Due to this, while command FSM is in “Wait for CCS state”or “Send CCSD”or “Boot Mode”, the Status register indicates status as 0 for the bit field [7:4].</p>
3	RO	0x0	fifo_full FIFO is full status
2	RO	0x1	fifo_empty FIFO is empty status
1	RO	0x1	fifo_tx_watermark FIFO reached Transmit watermark level; not qualified with data transfer
0	RO	0x0	fifo_rx_watermark FIFO reached Receive watermark level; not qualified with data transfer

**SDMMC\_FIFOTH**

Address: Operational Base + offset (0x004c)

FIFO threshold register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved
30:28	RW	0x0	<p>dma_multiple_transaction_size Burst size of multiple transaction; should be programmed same as DMA controller multiple-transaction-size SRC/DEST_MSIZE.</p> <p>3'b000: 1 transfers 3'b001: 4 3'b010: 8 3'b011: 16 3'b100: 32 3'b101: 64 3'b110: 128 3'b111: 256</p> <p>The unit for transfer is the H_DATA_WIDTH parameter. A single transfer (dw_dma_single assertion in case of Non DW DMA interface) would be signalled based on this value.</p> <p>Value should be sub-multiple of (RX_WMark + 1)* (F_DATA_WIDTH/H_DATA_WIDTH) and (FIFO_DEPTH - TX_WMark)* (F_DATA_WIDTH/ H_DATA_WIDTH)</p> <p>For example, if FIFO_DEPTH = 16, FDATA_WIDTH == H_DATA_WIDTH</p> <p>Allowed combinations for MSize and TX_WMark are:</p> <ul style="list-style-type: none"> <li>MSize = 1, TX_WMARK = 1-15</li> <li>MSize = 4, TX_WMark = 8</li> <li>MSize = 4, TX_WMark = 4</li> <li>MSize = 4, TX_WMark = 12</li> <li>MSize = 8, TX_WMark = 8</li> <li>MSize = 8, TX_WMark = 4</li> </ul> <p>Allowed combinations for MSize and RX_WMark are:</p> <ul style="list-style-type: none"> <li>MSize = 1, RX_WMARK = 0-14</li> <li>MSize = 4, RX_WMark = 3</li> <li>MSize = 4, RX_WMark = 7</li> <li>MSize = 4, RX_WMark = 11</li> <li>MSize = 8, RX_WMark = 7</li> </ul> <p>Recommended:</p> <p>MSize = 8, TX_WMark = 8, RX_WMark = 7</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27:16	RW	0x000	<p>rx_wmark FIFO threshold watermark level when receiving data to card. When FIFO data count reaches greater than this number, DMA/FIFO request is raised. During end of packet, request is generated regardless of threshold programming in order to complete any remaining data. In non-DMA mode, when receiver FIFO threshold (RXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, interrupt is not generated if threshold programming is larger than any remaining data. It is responsibility of host to read remaining bytes on seeing Data Transfer Done interrupt.</p> <p>In DMA mode, at end of packet, even if remaining bytes are less than threshold, DMA request does single transfers to flush out any remaining bytes before Data Transfer Done interrupt is set.</p> <p>12 bits-1 bit less than FIFO-count of status register, which is 13 bits.</p> <p>Limitation: RX_WMark &lt;= FIFO_DEPTH-2 Recommended: (FIFO_DEPTH/2) - 1; (means greater than (FIFO_DEPTH/2) - 1) NOTE: In DMA mode during CCS time-out, the DMA does not generate the request at the end of packet, even if remaining bytes are less than threshold. In this case, there will be some data left in the FIFO. It is the responsibility of the application to reset the FIFO after the CCS timeout.</p>
15:12	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:0	RW	0x000	<p>tx_wmark FIFO threshold watermark level when transmitting data to card.</p> <p>When FIFO data count is less than or equal to this number, DMA/FIFO request is raised. If Interrupt is enabled, then interrupt occurs.</p> <p>During end of packet, request or interrupt is generated, regardless of threshold programming.</p> <p>In non-DMA mode, when transmit FIFO threshold (TXDR) interrupt is enabled, then interrupt is generated instead of DMA request.</p> <p>During end of packet, on last interrupt, host is responsible for filling FIFO with only required remaining bytes (not before FIFO is full or after CIU completes data transfers, because FIFO may not be empty).</p> <p>In DMA mode, at end of packet, if last transfer is less than burst size, DMA controller does single cycles until required bytes are transferred.</p> <p>12 bits -1 bit less than FIFO-count of status register, which is 13 bits.</p> <p>Limitation: TX_WMark &gt;= 1;</p> <p>Recommended: FIFO_DEPTH/2; (means less than or equal to FIFO_DEPTH/2)</p>

**SDMMC\_CDETECT**

Address: Operational Base + offset (0x0050)

Card-detect register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RO	0x0	card_detect_n Value on card_detect_n input ports; read-only bits. 0 represents presence of card.

**SDMMC\_WRTPRT**

Address: Operational Base + offset (0x0054)

Write-protect register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	write_protect Value on card_write_prt input port. 1 represents write protection.

**SDMMC\_TCBCNT**

Address: Operational Base + offset (0x005c)

Transferred CIU card byte count

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	<p>trans_card_byte_count Number of bytes transferred by CIU unit to card.</p> <p>In 32-bit or 64-bit AMBA data-bus-width modes, register should be accessed in full to avoid read-coherency problems. In 16-bit AMBA data-bus-width mode, internal 16-bit coherency register is implemented. User should first read lower 16 bits and then higher 16 bits. When reading lower 16 bits, higher 16 bits of counter are stored in temporary register. When higher 16 bits are read, data from temporary register is supplied.</p> <p>Both TCBCNT and TBBCNT share same coherency register.</p> <p>When AREA_OPTIMIZED parameter is 1, register should be read only after data transfer completes; during data transfer, register returns 0.</p>

**SDMMC\_TBBCNT**

Address: Operational Base + offset (0x0060)

Transferred host/DMA to/from BIU-FIFO byte count

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	<p>trans_fifo_byte_count Number of bytes transferred between Host/DMA memory and BIU FIFO.</p> <p>In 32-bit or 64-bit AMBA data-bus-width modes, register should be accessed in full to avoid read-coherency problems. In 16-bit AMBA data-bus-width mode, internal 16-bit coherency register is implemented. User should first read lower 16 bits and then higher 16 bits. When reading lower 16 bits, higher 16 bits of counter are stored in temporary register. When higher 16 bits are read, data from temporary register is supplied.</p> <p>Both TCBCNT and TBBCNT share same coherency register.</p>

**SDMMC\_DEBNCE**

Address: Operational Base + offset (0x0064)

Card detect debounce register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23:0	RW	0xffffffff	debounce_count Number of host clocks (clk) used by debounce filter logic; typical debounce time is 5-25 ms.

**SDMMC\_USRID**

Address: Operational Base + offset (0x0068)

User ID register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x07967797	usrid User identification register; value set by user. Default reset value can be picked by user while configuring core before synthesis. Can also be used as scratch pad register by user. The default value is determined by Configuration Value.

**SDMMC\_VERID**

Address: Operational Base + offset (0x006c)

Synopsys version ID register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x5342270a	verid Version identification register; register value is hard-wired. Can be read by firmware to support different versions of core.

**SDMMC\_HCON**

Address: Operational Base + offset (0x0070)

Hardware Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	<p>HCON Configuration Dependent. Hardware configurations selected by user before synthesizing core. Register values can be used to develop configuration-independent software drivers.</p> <ul style="list-style-type: none"> <li>[0]: CARD_TYPE           <ul style="list-style-type: none"> <li>0: MMC_ONLY</li> <li>1: SD_MMC</li> </ul> </li> <li>[5:1]: NUM_CARDS - 1</li> <li>[6]: H_BUS_TYPE           <ul style="list-style-type: none"> <li>0: APB</li> <li>1: AHB</li> </ul> </li> <li>[9:7]: H_DATA_WIDTH           <ul style="list-style-type: none"> <li>000: 16 bits</li> <li>001: 32 bits</li> <li>010: 64 bits</li> <li>others: reserved</li> </ul> </li> <li>[15:10]: H_ADDR_WIDTH           <ul style="list-style-type: none"> <li>0 to 7: reserved</li> <li>8: 9 bits</li> <li>9: 10 bits</li> <li>...</li> </ul> </li> <li>31: 32 bits</li> <li>32 to 63: reserved</li> <li>[17:16]: DMA_INTERFACE           <ul style="list-style-type: none"> <li>00: none</li> <li>01: DW_DMA</li> <li>10: GENERIC_DMA</li> <li>11: NON-DW-DMA</li> </ul> </li> <li>[20:18]: GE_DMA_DATA_WIDTH           <ul style="list-style-type: none"> <li>000: 16 bits</li> <li>001: 32 bits</li> <li>010: 64 bits</li> <li>others: reserved</li> </ul> </li> <li>[21]: FIFO_RAM_INSIDE           <ul style="list-style-type: none"> <li>0: outside</li> <li>1: inside</li> </ul> </li> <li>[22]: IMPLEMENT_HOLD_REG           <ul style="list-style-type: none"> <li>0: no hold register</li> <li>1: hold register</li> </ul> </li> <li>[23]: SET_CLK_FALSE_PATH           <ul style="list-style-type: none"> <li>0: no false path</li> <li>1: false path set</li> </ul> </li> <li>[25:24]: NUM_CLK_DIVIDER-1</li> <li>[26]: AREA_OPTIMIZED           <ul style="list-style-type: none"> <li>0: no area optimization</li> <li>1: Area optimization</li> </ul> </li> </ul>

**SDMMC\_UHS\_REG**

Address: Operational Base + offset (0x0074)

UHS-1 register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved
16	RW	0x0	<p>ddr_reg</p> <p>DDR mode. Determines the voltage fed to the buffers by an external voltage regulator.</p> <p>0: non-DDR mode</p> <p>1: DDR mode</p> <p>UHS_REG [16] should be set for card.</p>
15:1	RO	0x0	reserved
0	RW	0x0	<p>volt_reg</p> <p>High Voltage mode. Determines the voltage fed to the buffers by an external voltage regulator.</p> <p>0: buffers supplied with 3.3V Vdd</p> <p>1: buffers supplied with 1.8V Vdd</p> <p>These bits function as the output of the host controller and are fed to an external voltage regulator. The voltage regulator must switch the voltage of the buffers of a particular card to either 3.3V or 1.8V, depending on the value programmed in the register.</p> <p>VOLT_REG[0] should be set to 1'b1 for card in order to make it operate for 1.8V.</p>

**SDMMC\_RST\_N**

Address: Operational Base + offset (0x0078)

Hardware reset register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x1	<p>card_reset</p> <p>Hardware reset.</p> <p>0: active mode</p> <p>1: reset</p> <p>These bits cause the cards to enter pre-idle state, which requires them to be re-initialized.</p> <p>CARD_RESET[0] should be set to 1'b1 to reset card.</p>

**SDMMC\_BMOD**

Address: Operational Base + offset (0x0080)

Bus Mode Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved
10:8	RO	0x0	<p>PBL Programmable Burst Length. These bits indicate the maximum number of beats to be performed in one IDMAC transaction. The IDMAC will always attempt to burst as specified in PBL each time it starts a Burst transfer on the host bus. The permissible values are 1, 4, 8, 16, 32, 64, 128 and 256. This value is the mirror of MSIZE of FIFOTH register. In order to change this value, write the required value to FIFOTH register. This is an encode value as follows.</p> <ul style="list-style-type: none"> <li>000 – 1 transfers</li> <li>001 – 4 transfers</li> <li>010 – 8 transfers</li> <li>011 – 16 transfers</li> <li>100 – 32 transfers</li> <li>101 – 64 transfers</li> <li>110 – 128 transfers</li> <li>111 – 256 transfers</li> </ul> <p>Transfer unit is either 16, 32, or 64 bits, based on HDATA_WIDTH. PBL is a read-only value and is applicable only for Data Access; it does not apply to descriptor accesses.</p>
7	RW	0x0	DE IDMAC Enable. When set, the IDMAC is enabled.
6:2	RW	0x00	DSL Descriptor Skip Length. Specifies the number of HWord/Word/Dword (depending on 16/32/64-bit bus) to skip between two unchained descriptors. This is applicable only for dual buffer structure.
1	RW	0x0	FB Fixed Burst. Controls whether the AHB Master interface performs fixed burst transfers or not. When set, the AHB will use only SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers. When reset, the AHB will use SINGLE and INCR burst transfer operations.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	SWR Software Reset. When set, the DMA Controller resets all its internal registers It is automatically cleared after 1 clock cycle

**SDMMC\_PLDMND**

Address: Operational Base + offset (0x0084)

Poll Demand Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	WO	0x00000000	PD Poll Demand. If the OWN bit of a descriptor is not set, the FSM goes to the Suspend state. The host needs to write any value into this register for the IDMAC FSM to resume normal descriptor fetch operation. This is a write only register.

**SDMMC\_DBADDR**

Address: Operational Base + offset (0x0088)

Descriptor List Base Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	SDL Start of Descriptor List. Contains the base address of the First Descriptor. The LSB bits [0/1/2:0] for 16/32/64-bit bus-width) are ignored and taken as all-zero by the IDMAC internally. Hence these LSB bits are read-only.

**SDMMC\_IDSTS**

Address: Operational Base + offset (0x008c)

Internal DMAC Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16:13	RO	0x0	<p>FSM</p> <p>DMAC FSM present state.</p> <ul style="list-style-type: none"> <li>0: DMA_IDLE</li> <li>1: DMA_SUSPEND</li> <li>2: DESC_RD</li> <li>3: DESC_CHK</li> <li>4: DMA_RD_REQ_WAI</li> <li>5: DMA_WR_REQ_WAI</li> <li>6: DMA_RD</li> <li>7: DMA_WR</li> <li>8: DESC_CLOSE</li> </ul>
12:10	RO	0x0	<p>EB</p> <p>Error Bits. Indicates the type of error that caused a Bus Error. Valid only with atal Bus Error bit—IDSTS[2] (IDSTS64[2], in case of 64-bit address configuration) set. This field does not generate an interrupt.</p> <ul style="list-style-type: none"> <li>1: Host Abort received during transmission</li> <li>2: Host Abort received during reception</li> <li>Others: Reserved</li> </ul>
9	RW	0x0	<p>AIS</p> <p>Abnormal Interrupt Summary. Logical OR of the following:</p> <ul style="list-style-type: none"> <li>IDSTS[2] Fatal Bus Interrupt</li> <li>IDSTS[4] DU bit Interrupt</li> </ul> <p>Only unmasked bits affect this bit.</p> <p>This is a sticky bit and must be cleared each time a corresponding bit that causes AIS to be set is cleared. Writing a 1 clears this bit.</p>
8	RW	0x0	<p>NIS</p> <p>Normal Interrupt Summary. Logical OR of the following:</p> <ul style="list-style-type: none"> <li>IDSTS[0] Transmit Interrupt</li> <li>IDSTS[1] Receive Interrupt</li> </ul> <p>Only unmasked bits affect this bit.</p> <p>This is a sticky bit and must be cleared each time a corresponding bit that causes NIS to be set is cleared. Writing a 1 clears this bit.</p>
7:6	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	CES Card Error Summary. Indicates the status of the transaction to/from the card; also present in RINTSTS. Indicates the logical OR of the following bits: <ul style="list-style-type: none"> <li>■ EBE –End Bit Error</li> <li>■ RTO –Response Timeout/Boot Ack Timeout</li> <li>■ RCRC –Response CRC</li> <li>■ SBE –Start Bit Error</li> <li>■ DRTO –Data Read Timeout/BDS timeout</li> <li>■ DCRC –Data CRC for Receive</li> <li>■ RE –Response Error</li> </ul> Writing a 1 clears this bit. The abort condition of the IDMAC depends on the setting of this CES bit. If the CES bit is enabled, then the IDMAC aborts on a “response error”; however, it will not abort if the CES bit is cleared
4	RW	0x0	DU Descriptor Unavailable Interrupt. This bit is set when the descriptor is unavailable due to OWN bit = 0 (DES0[31] = 0). Writing a 1 clears this bit.
3	RO	0x0	reserved
2	RW	0x0	FBE Fatal Bus Error Interrupt. Indicates that a Bus Error occurred (IDSTS[12:10]) (IDSTS64[12:10], in case of 64-bit address configuration). When this bit is set, the DMA disables all its bus accesses. Writing a 1 clears this bit.
1	RW	0x0	RI Receive Interrupt. Indicates the completion of data reception for a descriptor. Writing a 1 clears this bit.
0	RW	0x0	TI Transmit Interrupt. Indicates that data transmission is finished for a descriptor. Writing 1 clears this bit

**SDMMC\_IDINTEN**

Address: Operational Base + offset (0x0090)

Internal DMAC Interrupt Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	AI Abnormal Interrupt Summary Enable. When set, an abnormal interrupt is enabled. This bit enables the following bits: <ul style="list-style-type: none"> <li>■ IDINTEN[2] Fatal Bus Error Interrupt</li> <li>■ IDINTEN[4] DU Interrupt</li> </ul>
8	RW	0x0	NI Normal Interrupt Summary Enable. When set, a normal interrupt is enabled. When reset, a normal interrupt is disabled. This bit enables the following bits: <ul style="list-style-type: none"> <li>■ IDINTEN[0] Transmit Interrupt</li> <li>■ IDINTEN[1] Receive Interrup</li> </ul>
7:6	RO	0x0	reserved
5	RW	0x0	CES Card Error summary Interrupt Enable. When set, it enables the Card Interrupt summary
4	RW	0x0	DU Descriptor Unavailable Interrupt. When set along with Abnormal Interrupt Summary Enable, the DU interrupt is enabled
3	RO	0x0	reserved
2	RW	0x0	FBE Fatal Bus Error Enable. When set with Abnormal Interrupt Summary Enable, the Fatal Bus Error Interrupt is enabled. When reset, Fatal Bus Error Enable Interrupt is disabled.
1	RW	0x0	RI Receive Interrupt Enable. When set with Normal Interrupt Summary Enable, Receive Interrupt is enabled. When reset, Receive Interrupt is disabled
0	RW	0x0	TI Transmit Interrupt Enable. When set with Normal Interrupt Summary Enable, Transmit Interrupt is enabled. When reset, Transmit Interrupt is disabled.

**SDMMC\_DSCADDR**

Address: Operational Base + offset (0x0094)

Current Host Descriptor Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	HDA Host Descriptor Address Pointer. Cleared on reset. Pointer updated by IDMAC during operation. This register points to the start address of the current descriptor read by the IDMAC

**SDMMC\_BUADDR**

Address: Operational Base + offset (0x0098)

Current Buffer Descriptor Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	HBA Host Buffer Address Pointer. Cleared on Reset. Pointer updated by IDMAC during operation. This register points to the current Data Buffer Address being accessed by the IDMAC

**SDMMC\_CARDTHRCTL**

Address: Operational Base + offset (0x0100)

Card read threshold enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:16	RW	0x000	CardRdThreshold Card Read Threshold size
15:2	RO	0x0	reserved
1	RW	0x0	BsyClrIntEn Busy Clear Interrupt generation: 0: Busy Clear Interrupt disabled 1: Busy Clear Interrupt enabled Note: The application can disable this feature if it does not want to wait for a Busy Clear Interrupt. For example, in a multi-card scenario, the application can switch to the other card without waiting for a busy to be completed. In such cases, the application can use the polling method to determine the status of busy. By default this feature is disabled and backward-compatible to the legacy drivers where polling is used.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	CardRdThrEn Card Read Threshold Enable. 0: Card Read Threshold disabled 1: Card Read Threshold enabled. Host Controller initiates Read Transfer only if CardRdThreshold amount of space is available in receive FIFO.

**SDMMC\_BACK\_END\_POWER**

Address: Operational Base + offset (0x0104)

Back-end power

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	back_end_power Back end power 0: Off; Reset 1: Back-end Power supplied to card application

**SDMMC\_UHS\_REG\_EXT**

Address: Operational Base + offset (0x0108)

UHS Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	Reserved

**SDMMC\_EMMC\_DDR\_REG**

Address: Operational Base + offset (0x010c)

eMMC 4.5 DDR START Bit Detection Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	HALF_START_BIT Control for start bit detection mechanism inside DWC_mobile_storage based on duration of start bit; each bit refers to one slot. For eMMC 4.5, start bit can be: 0: Full cycle (HALF_START_BIT = 0) 1: Less than one full cycle (HALF_START_BIT = 1) Set HALF_START_BIT=1 for eMMC 4.5 and above; set to 0 for SD applications.

**SDMMC\_ENABLE\_SHIFT**

Address: Operational Base + offset (0x0110)

Enable Phase Shift Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	Reserved

### SDMMC\_FIFO\_BASE

Address: Operational Base + offset (0x0200)

FIFO Base Address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	fifo_base_addr FIFO base address.

## 14.5 Interface Description

The interface and IOMUX setting for SDMMC, SDIO0, SDIO1, EMMC are shown as follows.

### 14.5.1 SDMMC IOMUX

Table 14-8 IOMUX Settings for SDMMC

<b>Module Pin</b>	<b>Direction</b>	<b>Pad Name</b>	<b>IOMUX Setting</b>
sdmmc_cclk	O	IO_SDMMC0clkout_JTAGtdo_S DCARDgpio6c4	GRF_GPIO6C_IOMUX[9:8]=2'b01
sdmmc_ccmd	I/O	IO_SDMMC0cmd_SDCARDgpio 6c5	GRF_GPIO6C_IOMUX[11:10]=2'b01
sdmmc_cdata0	I/O	IO_SDMMC0data0_JTAGtms_S DCARDgpio6c0	GRF_GPIO6C_IOMUX[1:0]=2'b01
sdmmc_cdata1	I/O	IO_SDMMC0data1_JTAGtrstn_ SDCARDgpio6c1	GRF_GPIO6C_IOMUX[3:2]=2'b01
sdmmc_cdata2	I/O	IO_SDMMC0data2_JTAGtdi_S DCARDgpio6c2	GRF_GPIO6C_IOMUX[5:4]=2'b01
sdmmc_cdata3	I/O	IO_SDMMC0data3_JTAGtck_S DCARDgpio6c3	GRF_GPIO6C_IOMUX[7:6]=2'b01
sdmmc_cdetectn	I	IO_SDMMC0detectn_SDCARDgpi o6c6	GRF_GPIO6C_IOMUX[13:12]=2'b01

### 14.5.2 SDIO0 IOMUX

Table 14-9 IOMUX Settings for SDIO0

<b>Module Pin</b>	<b>Direction</b>	<b>Pad Name</b>	<b>IOMUX Setting</b>
sdio0_cclk	O	IO_SDIO0clkout_WIFIgpio4d1	GRF_GPIO4D_IOMUX[3:2]=2'b01
sdio0_ccmd	I/O	IO_SDIO0cmd_WIFIgpio4d0	GRF_GPIO4D_IOMUX[1:0]=2'b01
sdio0_cdata0	I/O	IO_SDIO0data0_WIFIgpio4c4	GRF_GPIO4C_IOMUX[9:8]=2'b01
sdio0_cdata1	I/O	IO_SDIO0data1_WIFIgpio4c5	GRF_GPIO4C_IOMUX[11:10]=2'b01
sdio0_cdata2	I/O	IO_SDIO0data2_WIFIgpio4c6	GRF_GPIO4C_IOMUX[13:12]=2'b01
sdio0_cdata3	I/O	IO_SDIO0data3_WIFIgpio4c7	GRF_GPIO4C_IOMUX[15:14]=2'b01
sdio0_cdetectn	I	IO_SDIO0detectn_WIFIgpio4d 2	GRF_GPIO4D_IOMUX[5:4]=2'b01

sdio0_wprt	I	IO_SDIO0wrprt_WIFIgpio4d3	GRF_GPIO4D_IOMUX[7:6]=2'b01
sdio0_int_n	I	IO_SDIO0intn_WIFIgpio4d6	GRF_GPIO4D_IOMUX[13:12]=2'b01
sdio0_pwren	O	IO_SDIO0pwren_WIFIgpio4d4	GRF_GPIO4D_IOMUX[9:8]=2'b01
sdio0_bkpwr	O	IO_SDIO0bkpwr_WIFIgpio4d5	GRF_GPIO4D_IOMUX[11:10]=2'b01

### 14.5.3 SDIO1 IOMUX

Table 14-10 IOMUX Settings for SDIO1

Module Pin	Direction	Pad Name	IOMUX Setting
sdio1_cclk	O	IO_FLASH1csn1_HOSTdout13 _MACcrs_SDIO1clkout_FLASH 1gpio4a7	GRF_GPIO4AH_IOMUX[15:12]=4'h4
sdio1_ccmd	I/O	IO_FLASH1csn0_HOSTdout12 _MACrxclk_SDIO1cmd_FLASH 1gpio4a6	GRF_GPIO4AH_IOMUX[11:8]=4'h4
sdio1_cdata0	I/O	IO_FLASH1data0_HOSTdout0 _MACtxd2_SDIO1data0_FLAS H1gpio3d0	GRF_GPIO3DL_IOMUX[3:0]=4'h4
sdio1_cdata1	I/O	IO_FLASH1data1_HOSTdout1 _MACtxd3_SDIO1data1_FLAS H1gpio3d1	GRF_GPIO3DL_IOMUX[7:4]=4'h4
sdio1_cdata2	I/O	IO_FLASH1data2_HOSTdout2 _MACrxid2_SDIO1data2_FLAS H1gpio3d2	GRF_GPIO3DL_IOMUX[11:8]=4'h4
sdio1_cdata3	I/O	IO_FLASH1data3_HOSTdout3 _MACrxid3_SDIO1data3_FLAS H1gpio3d3	GRF_GPIO3DL_IOMUX[15:12]=4'h4
sdio1_cdetectn	I	IO_FLASH1data4_HOSTdout4 _MACtxd0_SDIO1detectn_FLA SH1gpio3d4	GRF_GPIO3DH_IOMUX[3:0]=4'h4
sdio1_wprt	I	IO_FLASH1data5_HOSTdout5 _MACtxd1_SDIO1wrprt_FLAS H1gpio3d5	GRF_GPIO3DH_IOMUX[7:4]=4'h4
sdio1_int_n	I	IO_FLASH1data7_HOSTdout7 _MACrxid1_SDIO1intn_FLASH1 gpio3d7	GRF_GPIO3DH_IOMUX[15:12]=4'h4
sdio1_pwren	O	IO_FLASH1csn2_HOSTdout15 _MACtxclk_SDIO1pwren_FLAS H1gpio4b1	GRF_GPIO4BL_IOMUX[7:4]=4'h4
sdio1_bkpwr	O	IO_FLASH1data6_HOSTdout6 _MACrxid0_SDIO1bkpwr_FLAS H1gpio3d6	GRF_GPIO3DH_IOMUX[11:8]=4'h4

## 14.5.4 EMMC IOMUX

Table 14-11 IOMUX Settings for eMMC

Module Pin	Direction	Pad Name	IOMUX Setting
emmc_cclk	O	IO_FLASH0dqs_EMMCclkout_F LASH0gpio3c2	GRF_GPIO3C_IOMUX[5:4]=2'b10
emmc_ccmd	I/O	IO_FLASH0csn2_EMMCcmd_FL ASH0gpio3c0	GRF_GPIO3C_IOMUX[1:0]=2'b10
emmc_cdata0	I/O	IO_FLASH0data0_EMMCdata0 _FLASH0gpio3a0	GRF_GPIO3A_IOMUX[1:0]=2'b10
emmc_cdata1	I/O	IO_FLASH0data1_EMMCdata1 _FLASH0gpio3a1	GRF_GPIO3A_IOMUX[3:2]=2'b10
emmc_cdata2	I/O	IO_FLASH0data2_EMMCdata2 _FLASH0gpio3a2	GRF_GPIO3A_IOMUX[5:4]=2'b10
emmc_cdata3	I/O	IO_FLASH0data3_EMMCdata3 _FLASH0gpio3a3	GRF_GPIO3A_IOMUX[7:6]=2'b10
emmc_cdata4	I/O	IO_FLASH0data4_EMMCdata4 _FLASH0gpio3a4	GRF_GPIO3A_IOMUX[9:8]=2'b10
emmc_cdata5	I/O	IO_FLASH0data5_EMMCdata5 _FLASH0gpio3a5	GRF_GPIO3A_IOMUX[11:10]=2'b10
emmc_cdata6	I/O	IO_FLASH0data6_EMMCdata6 _FLASH0gpio3a6	GRF_GPIO3A_IOMUX[13:12]=2'b10
emmc_cdata7	I/O	IO_FLASH0data7_EMMCdata7 _FLASH0gpio3a7	GRF_GPIO3A_IOMUX[15:14]=2'b10
emmc_rstn	O	IO_FLASH0csn3_EMMCrstnout _FLASH0gpio3c1	GRF_GPIO3C_IOMUX[3:2]=2'b10
emmc_pwren	O	IO_FLASH0wp_EMMCpwren_F LASH0gpio3b1	GRF_GPIO3B_IOMUX[3:2]=2'b10

Notes: Direction: **I**- Input, **O**- Output, **I/O**- Input/Output

## 14.6 Application Notes

### 14.6.1 Card-Detect and Write-Protect Mechanism

Following figure illustrates how the SD/MMC card detection and write-protect signals are connected. Most of the SD/MMC sockets have card-detect pins. When no card is present, card\_detect\_n is 1 due to the pull-up. When the card is inserted, the card-detect pin is shorted to ground, which makes card\_detect\_n go to 0. Similarly in SD cards, when the write-protect switch is toward the left, it shorts the write\_protect port to ground.

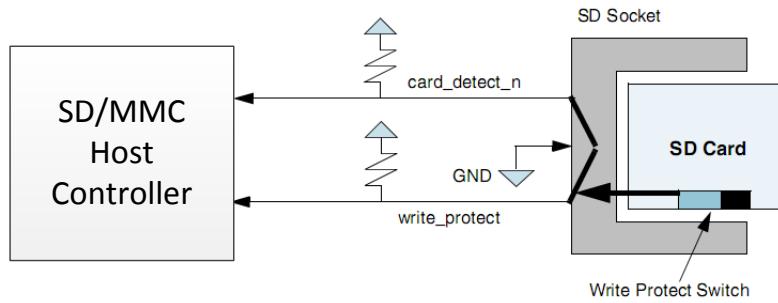


Fig. 14-9 SD/MMC Card-Detect and Write-Protect

## 14.6.2 SD/MMC Termination Requirement

Following Figure illustrates the SD/MMC termination requirements, which is required to pull up ccmd and cdata lines on the device bus. The recommended specification for pull-up on the ccmd line ( $R_{CMD}$ ) is 4.7K - 100K for MMC, and 10K - 100K for an SD. The recommended pull-up on the cdata line ( $R_{DAT}$ ) is 50K - 100K.

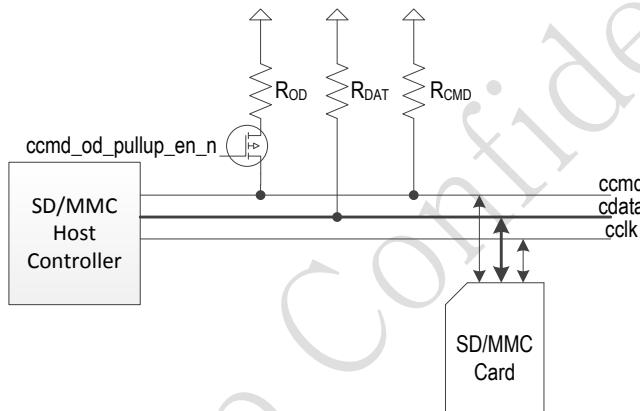


Fig. 14-10 SD/MMC Card Termination

### 1. $R_{CMD}$ and $R_{OD}$ Calculation

The SD/MMC card enumeration happens at a very low frequency – 100-400KHz. Since the MMC bus is a shared bus between multiple cards, during enumeration open-drive mode is used to avoid bus conflict. Cards that drive 0 win over cards that drive “z”. The pull-up in the command line pulls the bus to 1 when all cards drive “z”. During normal data transfer, the host chooses only one card and the card driver switches to push-pull mode.

For example, if enumeration is done at 400KHz and the total bus capacitance is 200 pf, the pull-up needed during enumeration is:

$$\begin{aligned} 2.2 \text{RC} &= \text{rise-time} = 1/400\text{KHz} \\ R &= 1/(2.2 * C * 100\text{KHz}) \\ &= 1/(2.2 * 200 * 10^{12} * 400 * 10^3) \\ &= 1/(17.6 * 10^{-5}) \\ &= 5.68\text{K} \end{aligned}$$

The  $R_{OD}$  and  $R_{CMD}$  should be adjusted in such a way that the effective pull-up is at the maximum 5.68K during enumeration. If there are only a few cards in the bus, a fixed  $R_{CMD}$  resistor is sufficient and there is no need for an additional  $R_{OD}$  pull-up during enumeration. You should also ensure the effective pull-up will not violate the I<sub>OL</sub> rating of the drivers.

In SD mode, since each card has a separate bus, the capacitance is less, typically in the order of 20-30pf (host capacitance + card capacitance + trace + socket capacitance). For example, if enumeration is done at 400KHz and the total bus capacitance is 20pf, the pull-up needed during enumeration is:

$$\begin{aligned} 2.2 \text{ RC} &= \text{rise-time} = 1/400\text{KHz} \\ R &= 1/(2.2 * C * 100\text{KHz}) \\ &= 1/(2.2 \times 20 \times 10^{**-12} \times 400 \times 10^{**3}) \\ &= 1/(1.76 \times 10^{**-5}) \\ &= 56.8\text{K} \end{aligned}$$

Therefore, a fixed 56.8K permanent Rcmd is sufficient in SD mode to enumerate the cards.

The driver of the SD/MMC on the "command" port needs to be only a push-pull driver. During enumeration, the SD/MMC emulates an open-drain driver by driving only a 0 or a "z" by controlling the ccmd\_out and ccmd\_out\_en signals.

#### 14.6.3 Software/Hardware Restriction

Before issuing a new data transfer command, the software should ensure that the card is not busy due to any previous data transfer command. Before changing the card clock frequency, the software must ensure that there are no data or command transfers in progress.

If the card is enumerated in SDR50, or DDR50 mode, then the application must program the use\_hold\_reg bit[29] in the CMD register to 1'b0 (phase shift of cclk\_in\_drv = 0) or 1'b1 (phase shift of cclk\_in\_drv>0). If the card is enumerated in SDR12 or SDR25 mode, the application must program the use\_hold\_reg bit[29] in the CMD register to 1'b1.

This programming should be done for all data transfer commands and non-data commands that are sent to the card. When the use\_hold\_reg bit is programmed to 1'b0, the Host Controller bypasses the Hold Registers in the transmit path. The value of this bit should not be changed when a Command or Data Transfer is in progress. For more details on using use\_hold\_reg and the implementation requirements for meeting the Card input hold time, refer to "Recommended Usage" in following table.

Table 14-12 Recommended Usage of use\_hold\_reg

No.	Speed Mode	use_hold_reg	cclk_in (MHz)	clk_in_drv (MHz)	clk_divider	Phase shift
1	SDR104	1'b0	200	200	0	0
2	SDR104	1'b1	200	200	0	Tunable> 0
3	SDR50	1'b0	100	100	0	0
4	SDR50	1'b1	100	100	0	Tunable> 0
5	DDR50 (8bit)	1'b0	100	100	1	0
6	DDR50 (8bit)	1'b1	100	100	1	Tunable> 0
7	DDR50 (4bit)	1'b0	50	50	0	0
8	DDR50 (4bit)	1'b1	50	50	0	Tunable> 0
9	SDR25	1'b1	50	50	0	Tunable> 0
10	SDR12	1'b1	50	50	1	Tunable> 0

To avoid glitches in the card clock outputs, the software should use the following steps when changing the card clock frequency:

- 1) Before disable the clocks, ensure that the card is not busy due to any previous data command. To determine this, check for 0 in bit9 of STATUS register.
- 2) Update the Clock Enable register to disable all clocks. To ensure completion of any previous command before this update, send a command to the CIU to update the clock registers by setting:
  - start\_cmd bit
  - “update clock registers only” bits
  - “wait\_previous data complete” bit

Wait for the CIU to take the command by polling for 0 on the start\_cmd bit.

- 3) Set the start\_cmd bit to update the Clock Divider and/or Clock Source registers, and send a command to the CIU in order to update the clock registers; wait for the CIU to take the command.
- 4) Set start\_cmd to update the Clock Enable register in order to enable the required clocks and send a command to the CIU to update the clock registers; wait for the CIU to take the command.

In non-DMA mode, while reading from a card, the Data Transfer Over (RINTSTS[3]) interrupt occurs as soon as the data transfer from the card is over. There still could be some data left in the FIFO, and the RX\_WMark interrupt may or may not occur, depending on the remaining bytes in the FIFO. Software should read any remaining bytes upon seeing the Data Transfer Over (DTO) interrupt. While using the external DMA interface for reading from a card, the DTO interrupt occurs only after all the data is flushed to memory by the DMA interface unit.

While writing to a card in external DMA mode, if an undefined-length transfer is selected by setting the Byte Count Register to 0, the DMA logic will likely request more data than it will send to the card, since it has no way of knowing at which point the software will stop the transfer. The DMA request stops as soon as the DTO is set by the CIU.

If the software issues a controller\_reset command by setting control register bit[0] to 1, all the CIU state machines are reset; the FIFO is not cleared. The DMA sends all remaining bytes to the host. In addition to a card-reset, if a FIFO reset is also issued, then:

- Any pending DMA transfer on the bus completes correctly
- DMA data read is ignored
- Write data is unknown(x)

Additionally, if dma\_reset is also issued, any pending DMA transfer is abruptly terminated. When the DW-DMA is used, the DMA controller channel should also be reset and reprogrammed.

If any of the previous data commands do not properly terminate, then the software should issue the FIFO reset in order to remove any residual data, if any, in the FIFO. After asserting the FIFO reset, you should wait until this bit is cleared.

One data-transfer requirement between the FIFO and host is that the number of transfers should be a multiple of the FIFO data width (32bits). For example, you want to write only 15 bytes to an SD/MMC card (BYTCNT), the host should write 16 bytes to the FIFO or program the DMA to do 16-byte transfers. The software can still program the Byte Count register to only 15, at which point only 15 bytes will be transferred to the card. Similarly, when 15 bytes are read from a card, the host should still read all 16 bytes from the FIFO.

It is recommended that you not change the FIFO threshold register in the middle of data transfers.

## 14.6.4 Programming Sequence

### 1. Initialization

Following figure illustrates the initialization flow.

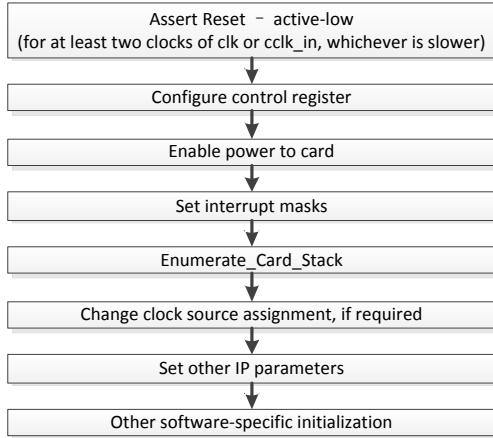


Fig. 14-11 Host Controller Initialization Sequence

Once the power and clocks are stable, reset\_n should be asserted(active-low) for at least two clocks of clk or cclk\_in, whichever is slower. The reset initializes the registers, ports, FIFO-pointers, DMA interface controls, and state-machines in the design. After power-on reset, the software should do the following:

- 1) Configure control register – For MMC mode, enable the open-drain pullup by setting enable\_OD\_pullup(bit24) in the control register.
- 2) Enable power to cards – Before enabling the power, confirm that the voltage setting to the voltage regulators is correct. Enable power to the connected cards by setting the corresponding bit to 1 in the Power Enable register. Wait for the power ramp-up time.
- 3) Set masks for interrupts by clearing appropriate bits in the Interrupt Mask register. Set the global int\_enable bit of the Control register. It is recommended that you write 0xffff\_ffff to the Raw Interrupt register in order to clear any pending interrupts before setting the int\_enable bit.
- 4) Enumerate card stack – Each card is enumerated according to card type; for details, refer to “Enumerated Card Stack”. For enumeration, you should restrict the clock frequency to 400KHz.
- 5) Changing clock source assignment – set the card frequency using the clock-divider and clock-source registers; for details, refer to “Clock Programming”. MMC cards operate at a maximum of 20MHz (at maximum of 52MHz in high-speed mode). SD mode operates at a maximum of 25MHz (at maximum of 50MHz in high-speed mode).
- 6) Set other parameters, which normally do not need to be changed with every command, with a typical value such as timeout values in cclk\_out according to SD/MMC specifications.
  - ResponseTimeOut = 0x64
  - DataTimeOut = highest of one of the following:  

$$(10*((TAAC*Fop)+(100*NSAC)))$$
 Host FIFO read/write latency from FIFO empty/full
  - Set the debounce value to 25ms(default:0xffff) in host clock cycle units in the DEBNCE register.
  - FIFO threshold value in bytes in the FIFOTH register.

### 2. Enumerated Card Stack

The card stack does the following:

- Enumerates all connected cards

- Sets the RCA for the connected cards
- Reads card-specific information
- Stores card-specific information locally

Enumeration depends on the operating mode of the SD/MMC card; the card type is first identified and the appropriate card enumeration routine is called.

- 1) Check if the card is connected.
- 2) Clear the card type register to set the card width as a single bit. For the given card number, clear the corresponding bits in the card\_type register. Clear the register bit for a 1-bit, 4-bit bus width. For example, for card number=1, clear bit 0 and bit 16 of the card\_type register.
- 3) Set clock frequency to  $F_{OD}=400\text{KHz}$ , maximum – Program clock divider0 (bits 0-7 in the CLKDIV register) value to one-half of the cclk\_in frequency divided by 400KHz. For example, if cclk\_in is 20MHz, then the value is  $20,000/(2*400)=25$ .
- 4) Identify the card type; that is, SD, MMC, or SDIO.
  - a. Send CMD5 first. If a response is received, then the card is SDIO
  - b. If not, send CMD8 with the following Argument
 

```
Bit[31:12] = 20'h0 //reserved bits
Bit[11:8] = 4'b0001 //VHS value
Bit[7:0] = 8'b10101010 //Preferred Check Pattern by SD2.0
```
  - c. If Response is received the card supports High Capacity SD2.0 then send ACMD41 with the following Argument
 

```
Bit[31] = 1'b0; //Reserved bits
Bit[30] = 1'b1; //High Capacity Status
Bit[29:24] = 6'h0; //Reserved bits
Bit[23:0] = Supported Voltage Range
```
  - d. If Response is received for ACMD41 then the card is SD. Otherwise the card is MMC.
  - e. If response is not received for initial CMD8 then card does not support High Capacity SD2.0, then issue CMD0 followed by ACMD41 with the following Argument
 

```
Bit[31] = 1'b0; //Reserved bits
Bit[30] = 1'b0; //High Capacity Status
Bit[29:24] = 6'h0; //Reserved bits
Bit[23:0] = Supported Voltage Range
```
- 5) Enumerate the card according to the card type.
- 6) Use a clock source with a frequency = Fod (that is, 400KHz) and use the following enumeration command sequence:
  - SD card – Send CMD0, CMD8, ACMD41, CMD2, CMD3.
  - MMC – Send CMD0, CMD1, CMD2, CMD3.

### 3. Power Control

You can implement power control using the following registers, along with external circuitry:

- Control register bits card\_voltage\_a and card\_voltage\_b – Status of these bits is reflected at the IO pins. The bits can be used to generate or control the supply voltage that the memory cards require.
- Power enable register – Control power to individual cards.

Programming these two register depends on the implemented external circuitry. While turning on or off the power enable, you should confirm that power supply settings are correct. Power to all cards usually should be disabled while switching off the power.

#### 4. Clock Programming

The Host Controller supports one clock sources. The clock to an individual card can be enabled or disabled. Registers that support this are:

- CLKDIV – Programs individual clock source frequency. CLKDIV limited to 0 or 1 is recommended.
- CLKSRC – Assign clock source for each card.
- CLKENA – Enables or disables clock for individual card and enables low-power mode, which automatically stops the clock to a card when the card is idle for more than 8 clocks.

The Host Controller loads each of these registers only when the start\_cmd bit and the Update\_clk\_regs\_only bit in the CMD register are set. When a command is successfully loaded, the Host Controller clears this bit, unless the Host Controller already has another command in the queue, at which point it gives an HLE(Hardware Locked Error).

Software should look for the start\_cmd and the Update\_clk\_regs\_only bits, and should also set the wait\_prvdata\_complete bit to ensure that clock parameters do not change during data transfer. Note that even though start\_cmd is set for updating clock registers, the Host Controller does not raise a command\_done signal upon command completion.

The following shows how to program these registers:

- 1) Confirm that no card is engaged in any transaction; if there is a transaction, wait until it finishes.
- 2) Stop all clocks by writing xxxx0000 to the CLKENA register. Set the start\_cmd, Update\_clk\_regs\_only, and wait\_prvdata\_complete bits in the CMD register. Wait until start\_cmd is cleared or an HLE is set; in case of an HLE, repeat the command.
- 3) Program the CLKDIV and CLKSRC registers, as required. Set the start\_cmd, Update\_clk\_regs\_only, and wait\_prvdata\_complete bits in the CMD register. Wait until start\_cmd is cleared or an HLE is set; in case of an HLE, repeat the command.
- 4) Re-enable all clocks by programming the CLKENA register. Set the start\_cmd, Update\_clk\_regs\_only, and wait\_prvdata\_complete bits in the CMD register. Wait until start\_cmd is cleared or an HLE is set; in case of an HLE, repeat the command.

#### 5. No-Data Command With or Without Response Sequence

To send any non-data command, the software needs to program the CMD register @0x2C and the CMDARG register @0x28 with appropriate parameters. Using these two registers, the Host Controller forms the command and sends it to the command bus. The Host Controller reflects the errors in the command response through the error bits of the RINTSTS register.

When a response is received – either erroneous or valid – the Host Controller sets the command\_done bit in the RINTSTS register. A short response is copied in Response Register0, while long response is copied to all four response registers @0x30, 0x34, 0x38, and 0x3C. The Response3 register bit 31 represents the MSB, and the Response0 register bit 0 represents the LSB of a long response.

For basic commands or non-data commands, follow these steps:

- 1) Program the Command register @0x28 with the appropriate command argument parameter.
- 2) Program the Command register @0x2C with the settings in following table.

Table 14-13 Command Settings for No-Data Command

Parameter	Value	Description
Default		

start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used; ref to "use_hold_reg" on CMD register
update_clk_regs_only	0	No clock parameters update command
data_expected	0	No data command
card number	0	Actual card number (one controller only connect one card, the num is No. 0)
cmd_index	command-index	-
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
<b>User-selectable</b>		
wait_prvdata_complete	1	Before sending command on command line, host should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
check_response_crc	1	If host should crosscheck CRC of response received

- 3) Wait for command acceptance by host. The following happens when the command is loaded into the Host Controller:
  - Host Controller accepts the command for execution and clears the start\_cmd bit in the CMD register, unless one command is in process, at which point the Host Controller can load and keep the second command in the buffer.
  - If the Host Controller is unable to load the command – that is, a command is already in progress, a second command is in the buffer, and a third command is attempted – then it generates an HLE (hardware-locked error).
- 4) Check if there is an HLE.
- 5) Wait for command execution to complete. After receiving either a response from a card or response timeout, the Host Controller sets the command\_done bit in the RINTSTS register. Software can either poll for this bit or respond to a generated interrupt.
- 6) Check if response\_timeout error, response\_CRC error, or response error is set. This can be done either by responding to an interrupt raised by these errors or by polling bits 1, 6, and 8 from the RINTSTS register @0x44. If no response error is received, then the response is valid. If required, the software can copy the response from the response registers @0x30-0x3C.

Software should not modify clock parameters while a command is being executed.

## 6. Data Transfer Commands

Data transfer commands transfer data between the memory card and the Host Controller. To send a data command, the Host Controller needs a command argument, total data size, and block size. Software can receive or send data through the FIFO.

Before a data transfer command, software should confirm that the card is not busy and is in a transfer state, which can be done using the CMD13 and CMD7 commands, respectively.

For the data transfer commands, it is important that the same bus width that is programmed in the card should be set in the card type register @0x18.

The Host Controller generates an interrupt for different conditions during data transfer, which are reflected in the RINTSTS register @0x44 as:

- 1) Data\_Transfer\_Over (bit 3) – When data transfer is over or terminated. If there is a response timeout error, then the Host Controller does not attempt any data transfer and the “Data Transfer Over” bit is never set.
- 2) Transmit\_FIFO\_Data\_request (bit 4) – FIFO threshold for transmitting data was reached; software is expected to write data, if available, in FIFO.
- 3) Receive\_FIFO\_Data\_request (bit 5) – FIFO threshold for receiving data was reached; software is expected to read data from FIFO.
- 4) Data starvation by Host timeout (bit 10) – FIFO is empty during transmission or is full during reception. Unless software writes data for empty condition or reads data for full condition, the Host Controller cannot continue with data transfer. The clock to the card has been stopped.
- 5) Data read timeout error (bit 9) – Card has not sent data within the timeout period.
- 6) Data CRC error (bit 7) – CRC error occurred during data reception.
- 7) Start bit error (bit 13) – Start bit was not received during data reception.
- 8) End bit error (bit 15) – End bit was not received during data reception or for a write operation; a CRC error is indicated by the card.

Conditions 6, 7, and 8 indicate that the received data may have errors. If there was a response timeout, then no data transfer occurred.

## 7. Single-Block or Multiple-Block Read

Steps involved in a single-block or multiple-block read are:

- 1) Write the data size in bytes in the BYTCNT register @0x20.
- 2) Write the block size in bytes in the BLKSIZ register @0x1C. The Host Controller expects data from the card in blocks of size BLKSIZ each.
- 3) Program the CMDARG register @0x28 with the data address of the beginning of a data read.
- 4) Program the Command register with the parameters listed in following table. For SD and MMC cards, use CMD17 for a single-block read and CMD18 for a multiple-block read. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

Table 14-14 Command Setting for Single or Multiple-Block Read

Parameter	Value	Description
<b>Default</b>		
start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used; ref to “use_hold_reg” on CMD register
update_clk_regs_only	0	No clock parameters update command
card number	0	Actual card number (one controller only connect one card, the num is No.0)
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
send_auto_stop	0/1	-
transfer_mode	0	Block transfer
read_write	0	Read from card
data_expected	1	Data command
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
<b>User-selectable</b>		
cmd_index	command-index	-

wait_prvdata_complete	1	0- Sends command immediately 1- Sends command after previous data transfer ends
check_response_crc	1	0- Host Controller should not check response CRC 1- Host Controller should check response CRC

After writing to the CMD register, the Host Controller starts executing the command; when the command is sent to the bus, the command\_done interrupt is generated.

- Software should look for data error interrupts; that is, bits 7, 9, 13, and 15 of the RINTSTS register. If required, software can terminate the data transfer by sending a STOP command.
- Software should look for Receive\_FIFO\_Data\_request and/or data starvation by host timeout conditions. In both cases, the software should read data from the FIFO and make space in the FIFO for receiving more data.
- When a Data\_Transfer\_Over interrupt is received, the software should read the remaining data from the FIFO.

## 8. Single-Block or Multiple-Block Write

Steps involved in a single-block or multiple-block write are:

- 1) Write the data size in bytes in the BYTCNT register @0x20.
- 2) Write the block size in bytes in the BLKSIZ register @0x1C; the Host Controller sends data in blocks of size BLKSIZ each.
- 3) Program CMDARG register @0x28 with the data address to which data should be written.
- 4) Write data in the FIFO; it is usually best to start filling data the full depth of the FIFO.
- 5) Program the Command register with the parameters listed in following table.

Table 14-15 Command Settings for Single or Multiple-Block Write

Parameter	Value	Description
<b>Default</b>		
start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used; ref to "use_hold_reg" on CMD register
update_clk_regs_only	0	No clock parameters update command
card number	0	Actual card number (one controller only connect one card, the num is No. 0)
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
send_auto_stop	0/1	-
transfer_mode	0	Block transfer
read_write	1	Write to card
data_expected	1	Data command
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
<b>User-selectable</b>		
cmd_index	command-index	-
wait_prvdata_complete	1	0- Sends command immediately 1- Sends command after previous data transfer ends
check_response_crc	1	0- Host Controller should not check response CRC 1- Host Controller should check response CRC

After writing to the CMD register, Host Controller starts executing a command; when the command is sent to the bus, a command\_done interrupt is generated.

- Software should look for data error interrupts; that is, for bits 7, 9, and 15 of the RINTSTS register. If required,

- software can terminate the data transfer by sending the STOP command.
- Software should look for Transmit\_FIFO\_Data\_Request and/or timeout conditions from data starvation by the host. In both cases, the software should write data into the FIFO.
- When a Data\_Transfer\_Over interrupt is received, the data command is over. For an open-ended block transfer, if the byte count is 0, the software must send the STOP command. If the byte count is not 0, then upon completion of a transfer of a given number of bytes, the Host Controller should send the STOP command, if necessary. Completion of the AUTO-STOP command is reflected by the Auto\_command\_done interrupt – bit 14 of the RINTSTS register. A response to AUTO\_STOP is stored in RESP1 @0x34.

## 9. Stream Read

A stream read is like the block read mentioned in “Single-Block or Multiple-Block Read”, except for the following bits in the Command register:

```
transfer_mode = 1; //Stream transfer
```

```
cmd_index = CMD20;
```

A stream transfer is allowed for only a single-bit bus width.

## 10. Stream Write

A stream write is exactly like the block write mentioned in “Single-Block or Multiple-Block Write”, except for the following bits in the Command register:

```
transfer_mode = 1;//Stream transfer
```

```
cmd_index = CMD11;
```

In a stream transfer, if the byte count is 0, then the software must send the STOP command. If the byte count is not 0, then when a given number of bytes completes a transfer, the Host Controller sends the STOP command. Completion of this AUTO\_STOP command is reflected by the Auto\_command\_done interrupt. A response to an AUTO\_STOP is stored in the RESP1 register@0x34.

A stream transfer is allowed for only a single-bit bus width.

## 11. Packed Commands

In order to reduce overhead, read and write commands can be packed in groups of commands—either all read or all write—that transfer the data for all commands in the group in one transfer on the bus.

Packed commands can be of two types:

- Packed Write: CMD23 →CMD25
- Packed Read: CMD23 → CMD25 → CMD23 → CMD18

Packed commands are put in packets by the application software and are transparent to the core.

## 12. Sending Stop or Abort in Middle of Transfer

The STOP command can terminate a data transfer between a memory card and the Controller, while the ABORT command can terminate an I/O data transfer for only the SDIO\_IOONLY and SDIO\_COMBO cards.

- Send STOP command – Can be sent on the command line while a data transfer is in progress; this command can be sent at any time during a data transfer.

You can also use an additional setting for this command in order to set the Command register bits (5-0) to CMD12 and set bit 14 (stop\_abort\_cmd) to 1. If stop\_abort\_cmd is not set to 1, the Controller does not know that the user stopped a data transfer. Reset bit 13 of the Command register (wait\_prvdata\_complete) to 0 in order to make the Controller send the command at once, even though there is a data transfer in progress.

- Send ABORT command – Can be used with only an SDIO\_IOONLY or SDIO\_COMBO card. To abort the function that is transferring data, program the function number in ASx bits (CCCR register of card, address 0x06, bits (0-2) using CMD52.

### 13. Suspend or Resume Sequence

In an SDIO card, the data transfer between an I/O function and the Controller can be temporarily halted using the SUSPEND command; this may be required in order to perform a high-priority data transfer with another function. When desired, the data transfer can be resumed using the RESUME command.

The following functions can be implemented by programming the appropriate bits in the CCCR register (Function 0) of the SDIO card. To read from or write to the CCCR register, use the CMD52 command.

- SUSPEND data transfer – Non-data command
  - 1) Check if the SDIO card supports the SUSPEND/RESUME protocol; this can be done through the SBS bit in the CCCR register @0x08 of the card.
  - 2) Check if the data transfer for the required function number is in process; the function number that is currently active is reflected in bits 0-3 of the CCCR register @0x0D. Note that if the BS bit (address 0xc::bit 0) is 1, then only the function number given by the FSx bits is valid.
  - 3) To suspend the transfer, set BR (bit 2) of the CCCR register @0x0C.
  - 4) Poll for clear status of bits BR (bit 1) and BS (bit 0) of the CCCR @0x0C. The BS (Bus Status) bit is 1 when the currently-selected function is using the data bus; the BR (Bus Release) bit remains 1 until the bus release is complete. When the BR and BS bits are 0, the data transfer from the selected function has been suspended.
- RESUME data transfer – This is a data command
  - 1) Check that the card is not in a transfer state, which confirms that the bus is free for data transfer.
  - 2) If the card is in a disconnect state, select it using CMD7. The card status can be retrieved in response to CMD52/CMD53 commands.
  - 3) Check that a function to be resumed is ready for data transfer; this can be confirmed by reading the RFx flag in CCCR @0x0F. If RF = 1, then the function is ready for data transfer.
  - 4) To resume transfer, use CMD52 to write the function number at FSx bits (0-3) in the CCCR register @0x0D. Form the command argument for CMD52 and write it in CMDARG @0x28.
  - 5) Write the block size in the BLKSIZ register @0x1C; data will be transferred in units of this block size.
  - 6) Write the byte count in the BYTCNT register @0x20. This is the total size of the data; that is, the remaining bytes to be transferred. It is the responsibility of the software to handle the data.
  - 7) Program Command register; similar to a block transfer.
  - 8) When the Command register is programmed, the command is sent and the function resumes data transfer. Read the DF flag (Resume Data Flag). If it is 1, then the function has data for the transfer and will begin a data transfer as soon as the function or memory is resumed. If it is 0, then the function has no data for the transfer.
  - 9) If the DF flag is 0, then in case of a read, the Host Controller waits for data. After the data timeout period, it gives a data timeout error.

### 14. Read\_Wait Sequence

Read\_wait is used with only the SDIO card and can temporarily stall the data transfer—either from function or memory—and allow the host to send commands to any function within the SDIO device. The host can stall this transfer for as long as required. The Host Controller provides the facility to signal this stall transfer to the card. The steps for doing this are:

- 1) Check if the card supports the read\_wait facility; read SRW (bit 2) of the CCCR register @0x08. If this bit is 1, then all functions in the card support the read\_wait facility. Use CMD52 to read this bit.
- 2) If the card supports the read\_wait signal, then assert it by setting the read\_wait (bit 6) in the CTRL register @0x00.
- 3) Clear the read\_wait bit in the CTRL register.

## 15. Controller/DMA/FIFO Reset Usage

- Controller reset – Resets the controller by setting the controller\_reset bit (bit 0) in the CTRL register; this resets the CIU and state machines, and also resets the BIU-to-CIU interface. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.
- FIFO reset - Resets the FIFO by setting the fifo\_reset bit (bit 1) in the CTRL register; this resets the FIFO pointers and counters of the FIFO. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.

In external DMA transfer mode, even when the FIFO pointers are reset, if there is a DMA transfer in progress, it could push or pop data to or from the FIFO; the DMA itself completes correctly. In order to clear the FIFO, the software should issue an additional FIFO reset and clear any FIFO underrun or overrun errors in the RAWINTS register caused by the DMA transfers after the FIFO was reset.

## 16. Card Read Threshold

When an application needs to perform a Single or Multiple Block Read command, the application must program the CardThrCtl register with the appropriate Card Read Threshold size (CardRdThreshold) and set the Card Read Threshold Enable (CardRdThrEnable) bit to 1'b1. This additional programming ensures that the Host controller sends a Read Command only if there is space equal to the CardRDThreshold available in the Rx FIFO. This in turn ensures that the card clock is not stopped in the middle a block of data being transmitted from the card. The Card Read Threshold can be set to the block size of the transfer, which guarantees that there is a minimum of one block size of space in the RxFIFO before the controller enables the card clock. The Card Read Threshold is required when the Round Trip Delay is greater than 0.5cclk\_in period.

## 17. Error Handling

The Host Controller implements error checking; errors are reflected in the RAWINTS register@0x44 and can be communicated to the software through an interrupt, or the software can poll for these bits. Upon power-on, interrupts are disabled (int\_enable in the CTRL register is 0), and all the interrupts are masked (bits 0-31 of the INTMASK register; default is 0).

Error handling:

- Response and data timeout errors – For response timeout, software can retry the command. For data timeout, the Host Controller has not received the data start bit – either for the first block or the intermediate block – within the timeout period, so software can either retry the whole data transfer again or retry from a specified block onwards. By reading the contents of the TCBCNT later, the software can decide how many bytes remain

to be copied.

- Response errors – Set when an error is received during response reception. In this case, the response that copied in the response registers is invalid. Software can retry the command.
- Data errors – Set when error in data reception are observed; for example, data CRC, start bit not found, end bit not found, and so on. These errors could be set for any block-first block, intermediate block, or last block. On receipt of an error, the software can issue a STOP or ABORT command and retry the command for either whole data or partial data.
- Hardware locked error – Set when the Host Controller cannot load a command issued by software. When software sets the start\_cmd bit in the CMD register, the Host Controller tries to load the command. If the command buffer is already filled with a command, this error is raised. The software then has to reload the command.
- FIFO underrun/overrun error – If the FIFO is full and software tries to write data in the FIFO, then an overrun error is set. Conversely, if the FIFO is empty and the software tries to read data from the FIFO, an underrun error is set. Before reading or writing data in the FIFO, the software should read the fifo\_empty or fifo\_full bits in the Status register.
- Data starvation by host timeout – Raised when the Host Controller is waiting for software intervention to transfer the data to or from the FIFO, but the software does not transfer within the stipulated timeout period. Under this condition and when a read transfer is in process, the software should read data from the FIFO and create space for further data reception. When a transmit operation is in process, the software should fill data in the FIFO in order to start transferring data to the card.
- CRC Error on Command – If a CRC error is detected for a command, the CE-ATA device does not send a response, and a response timeout is expected from the Host Controller. The ATA layer is notified that an MMC transport layer error occurred.

*Notes: During a multiple-block data transfer, if a negative CRC status is received from the device, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register. It then continues further data transmission until all the bytes are transmitted.*

#### 14.6.5 Voltage Switching

The Host Controller supports SD 3.0 Ultra High Speed (UHS-1) and is capable of voltage switching in SD-mode, which can be applied to SD High-Capacity (SDHC) and SD Extended Capacity (SDXC) cards. UHS-1 supports only 4-bit mode.

SD 3.0 UHS-1 supports the following transfer speed modes for UHS-50 and/or UHS-104 cards:

- DS – default-speed up to 25MHz, 3.3V signaling
- HS – high-speed up to 50MHz, 3.3V signaling
- SDR12 – SDR up to SDR 25MHz, 1.8V signaling
- SDR25 – SDR up to 50MHz, 1.8V signaling
- SDR50 – SDR up to 100MHz, 1.8V signaling
- DDR50 – DDR up to 50MHz, 1.8V signaling

Voltage selection can be done in only SD mode. The first CMD0 selects the bus mode-either SD mode or SPI mode. The card must be in SD mode in order for 1.8V signaling mode to apply, during which time the card cannot be switched to SPI mode or 3.3V signaling without a power

cycle.

If the System BIOS in an embedded system already knows that it is connected to an SD 3.0 card, then the driver programs the Controller to initiate ACMD41. The software knows from the response of ACMD41 whether or not the card supports voltage switching to 1.8V.

- If bit 32 of ACMD41 response is 1'b1: card supports voltage switching and next command-CMD11-invokes voltage switching sequence. After CMD11 is started, the software must program the VOLT\_REG register in the CSR space with the appropriate card number.
- If bit 32 of ACMD41 response is 1'b0: card does not support voltage switching and CMD11 should not be started.

If the card and host controller accept voltage switching, then they support UHS-1 modes of data transfer. After the voltage switch to 1.8V, SDR12 is the default speed.

Since the UHS-1 can be used in only 4-bit mode, the software must start ACMD6 and change the card data width to 4-bit mode; ACMD6 is driven in any of the UHS-1 speeds. If the host wants to select the DDR mode of data transfer, then the software must program the DDR\_REG register in the CSR space with the appropriate card number.

To choose from any of the SDR or DDR modes, appropriate values should be programmed in the CLKDIV register.

## 1. Voltage Switch Operation

The Voltage Switch operation must be performed in SD mode only.

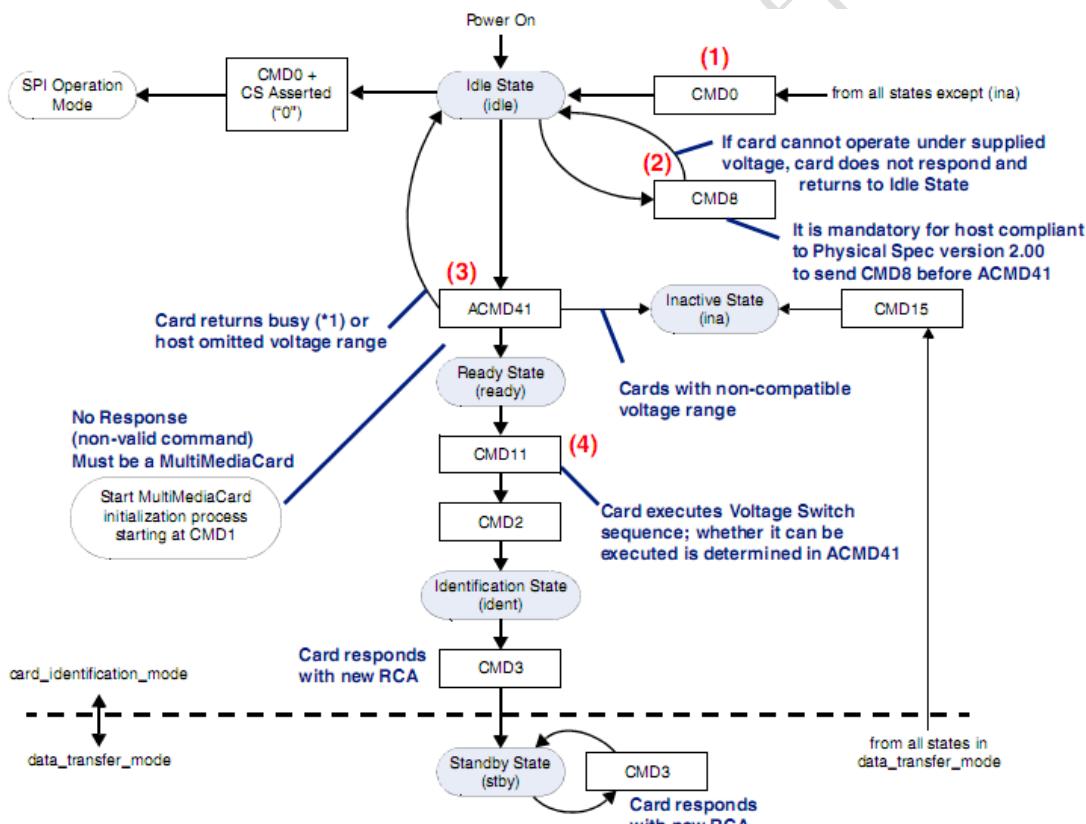


Fig. 14-12 Voltage Switching Command Flow Diagram

The following outlines the steps for the voltage switch programming sequence

- 1) Software Driver starts CMD0, which selects the bus mode as SD.
- 2) After the bus is in SD card mode, CMD8 is started in order to verify if the card is compatible with the SD Memory Card Specification, Version 2.00. CMD8 determines if the card is capable of working within the host

supply voltage specified in the VHS (19:16) field of the CMD; the card supports the current host voltage if a response to CMD8 is received.

- 3) ACMD 41 is started. The response to this command informs the software if the card supports voltage switching; bits 38, 36, and 32 are checked by the card argument of ACMD41; refer to following figure.

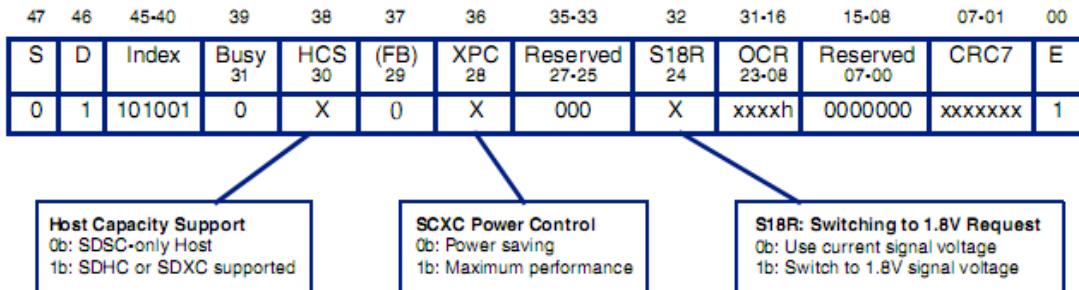


Fig. 14-13 ACMD41 Argument

- Bit 30 informs the card if host supports SDHC/SDXC or not; this bit should be set to 1'b1.
- Bit 28 can be either 1 or 0.
- Bit 24 should be set to 1'b1, indicating that the host is capable of voltage switching; refer to Figure 17-16.

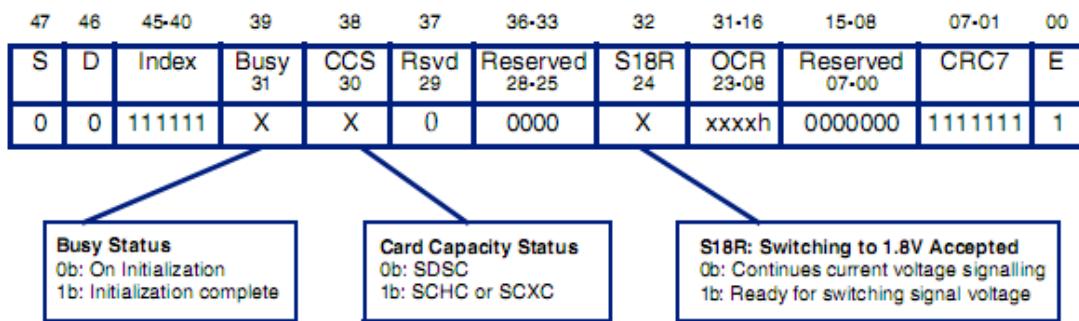


Fig. 14-14 ACMD41 Response(R3)

- Bit 30 – If set to 1'b1, card supports SDHC/SDXC; if set to 1'b0, card supports only SDSC
  - Bit 24 – If set to 1'b1, card supports voltage switching and is ready for the switch
  - Bit 31 – If set to 1'b1, initialization is over; if set to 1'b0, means initialization in process
- 4) If the card supports voltage switching, then the software must perform the steps discussed for either the “Voltage Switch Normal Scenario” or the “Voltage Switch Error Scenario”.

## 1. Voltage Switch Normal Scenario

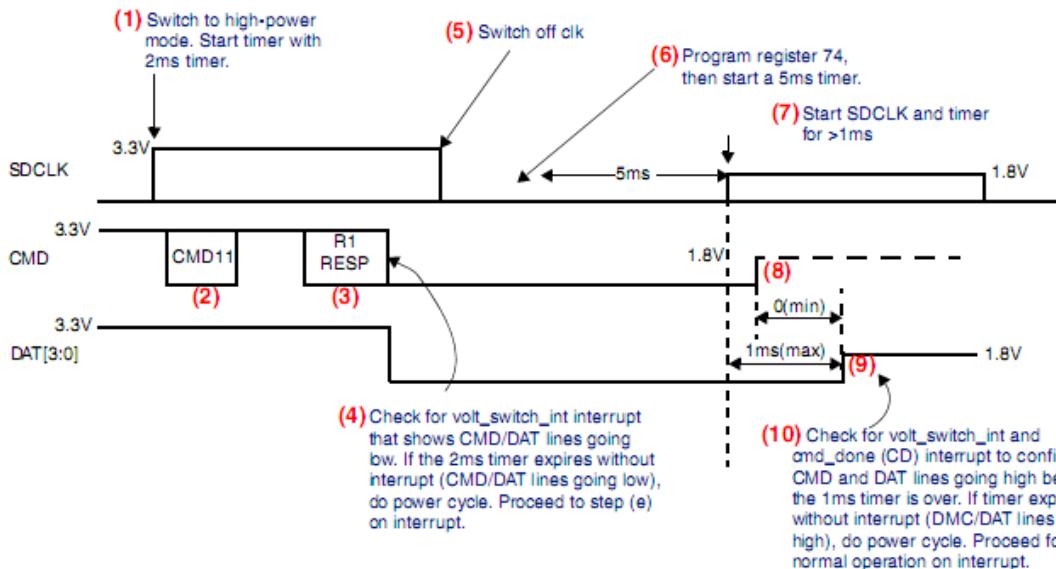


Fig. 14-15 Voltage Switch Normal Scenario

- The host programs CLKENA—cclk\_low\_power register—with zero (0) for the corresponding card, which makes the host controller move to high-power mode. The application should start a timer with a recommended value of 2ms; this value of 2 ms is determined as below: Total clk required for CMD11 = 48 clks

Total clk required for RESP R1 = 48 clks

Maximum clk delay between MCD11 end to start of RESP1 = 60 clks

Total = 48+48 + 60 = 160

Minimum frequency during enumeration is 100 KHz; that is, 10us

Total time = 160 \* 10us = 1600us = 1. 6ms ~ 2ms

- The host issues CMD11 to start the voltage switch sequence. Set bit 28 to 1'b1 in CMD when setting CMD11; for more information on setting bits, refer to “Boot Operation”.
- The card returns R1 response; the host controller does not generate cmd\_done interrupt on receiving R1 response.
- The card drives CMD and DAT [3:0] to low immediately after the response. The host controller generates interrupt (VOLT\_SWITCH\_INT) once the CMD or DAT [3:0] line goes low. The application should wait for this interrupt. If the 2ms timer expires without an interrupt (CMD/DAT lines going low), do a power cycle.

*Note: Before doing a power cycle, switch off the card clock by programming CLKENA register*

Proceed to step (5) on getting an interrupt (VOLT\_SWITCH\_INT).

*Note: This interrupt must be cleared once this interrupt is received. Additionally, this interrupt should not be masked during the voltage switch sequence.*

If the timer expires without interrupt (CMD/DAT lines going low), perform a power cycle.  
Proceed to step (5) on interrupt.

- Program the CLKENA, cclk\_enable register, with 0 for the corresponding card; the host stops supplying SDCLK.
- Program VOLT\_REG to the required values for the corresponding card. The application must program the newly-defined VOLT\_REG register to assign 1 for the bit corresponding to the card number. The application should start a timer > 5ms.
- After the 5ms timer expires, the host voltage regulator is stable. Program CLKENA, cclk\_enable register, with 1

for the corresponding card; the host starts providing SDCLK at 1.8V; this can be at zero time after VOLT\_REG has been programmed. When the CLKENA register is programmed, the application should start another timer > 1ms.

- 8) By detecting SDCLK, the card drives CMD to high at 1.8V for at least one clock and then stops driving (tri-state); CMD is triggered by the rising edge of SDCLK (SDR timing).
- 9) If switching to 1.8V signaling is completed successfully, the card drives DAT [3:0] to high at 1.8V for at least one clock and then stops driving (tri-state); DAT [3:0] is triggered by the rising edge of SDCLK (SDR timing). DAT[3:0] must be high within 1ms from the start of SDCLK.
- 10) The host controller generates a voltage switch interrupt (VOLT\_SWITCH\_INT) and a command done (CD) interrupt once the CMD and DAT[3:0] lines go high. The application should wait for this interrupt to confirm CMD and DAT lines going high before the 1ms timer is done.

If the timer expires without the voltage switch interrupt (VOLT\_SWITCH\_INT), a power cycle should be performed. Program the CLKENA register to stop the clock for the corresponding card number. Wait for the cmd\_done (CD) interrupt. Proceed for normal operation on interrupt. After the sequence is completed, the host and the card start communication in SDR12 timing.

## 2. Voltage Switch Error Scenario

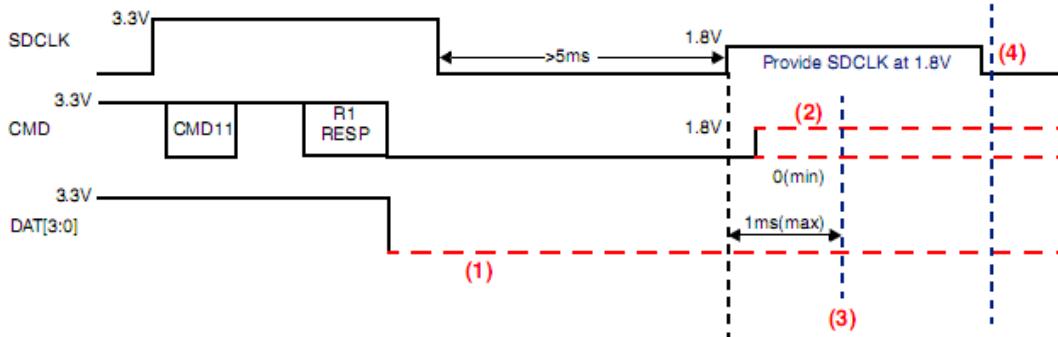


Fig. 14-16 Voltage Switch Error Scenario

- 1) If the interrupt (VOLT\_SWITCH\_INT) does not come, then the 2 ms timer should time out and a power cycle should be initiated.

*Note: Before performing a power cycle, switch off the card clock by programming CLKENA register; no cmd\_done (CD) interrupt is generated.*

Additionally, if the card detects a voltage error at any point in between steps (5) and (7) in Figure 17-17, the card keeps driving DAT[3:0] to low until card power off.

- 2) CMD can be low or tri-state.
- 3) The host controller generates a voltage switch interrupt once the CMD and DAT[3:0] lines go high. The application should check for an interrupt to confirm CMD and DAT lines going high before the 1 ms timer is done.

If the 1 ms timer expires without interrupt (VOLT\_SWITCH\_INT) and cmd\_done (CD), a power cycle should be performed. Program the CLKENA register to stop SDCLK of the corresponding card. Wait for the cmd\_done interrupt. Proceed for normal operation on interrupt.

- 4) If DAT[3:0] is low, the host drives SDCLK to low and then stops supplying the card power.

*Note: The card checks voltages of its own regulator output and host signals to ensure they are less than 2.5V. Errors are indicated by (1) and (2) in Figure 7-18.*

- If voltage switching is accepted by the card, the default speed is SDR12.
- Command Done is given:
  - If voltage switching is properly done, CMD and DAT line goes high.

- If switching is not complete, the 1ms timer expires, and the card clk is switched off.

*Note: No other CMD should be driven before the voltage switching operation is completed and Command Done is received.*

- The application should use CMD6 to check and select the particular function; the function appropriate-speed should be selected.

After the function switches, the application should program the correct value in the CLKDIV register, depending on the function chosen. Additionally, if Function 0x4 of the Access mode is chosen—that is, DDR50, then the application should also program 1'b1 in DDR\_REG for the card number that has been selected for DDR50 mode.

## 14.6.6 Back-End Power

Each device needs one bit to control the back-end power supply for an embedded device; this bit does not control the VDDH of the host controller. A back\_end\_power register enables software programming for back-end power. The value on this register is output to the back\_end\_power signal, which can be used to switch power on and off the embedded device.

## 14.6.7 DDR Operation

### 1. 4-bit DDR Programming Sequence

DDR programming should be done only after the voltage switch operation has completed. The following outlines the steps for the DDR programming sequence:

- 1) Once the voltage switch operation is complete, the user must program VOLT\_REG to the required values for the corresponding card.
  - To start a card to work in DDR mode, the application must program a bit of the newly defined VOLT\_REG[31:16] register with a value of 1'b1.
  - The bit that the user programs depends on which card is to be accessed in DDR mode.
- 2) To move back to SDR mode, a power cycle should be run on the card—putting the card in SDR12 mode—and only then should VOLT\_REG[31:16] be set back to 1'b0 for the appropriate card.

### 2. 8-bit DDR Programming Sequence

The following outlines the steps for the 8-bit DDR programming sequence:

- 1) The cclk\_in signal should be twice the speed of the required cclk\_out. Thus, if the cclk\_out signal is required to be 50 MHz, the cclk\_in signal should be 100 MHz.
- 2) The CLKDIV register should always be programmed with a value higher than zero (0); that is, a clock divider should always be used for 8-bit DDR mode.
- 3) The application must program the UHS\_REG [31:16] register (DDR\_REG bits) by assigning it with a value of 1 for the bit corresponding to the card number; this causes the selected card to start working in DDR mode.
- 4) Depending on the card number, the CTYPE [31:16] bits should be set in order to make the host work in the 8-bit mode.

### 3. eMMC4.5 DDR START Bit

The eMMC4.5 changes the START bit definition in the following manner:

- 1) Receiver samples the START bit on the rising edge.

- 2) On the next rising edge after sampling the START bit, the receiver must sample the data.
- 3) Removes requirement of the START bit and END bit to be high for one full cycle.

*Notes: The Host Controller does not support a START bit duration higher than one clock cycle. START bit durations of one or less than one clock cycle are supported and can be defined at the time of startup by programming the EMMC\_DDR\_REG register.*

Following figure illustrates cases for the definition change of the START bit with eMMC4.5; it also illustrates how some of these cases can fail in sampling when higher-value delays are considered for I/O PADs.

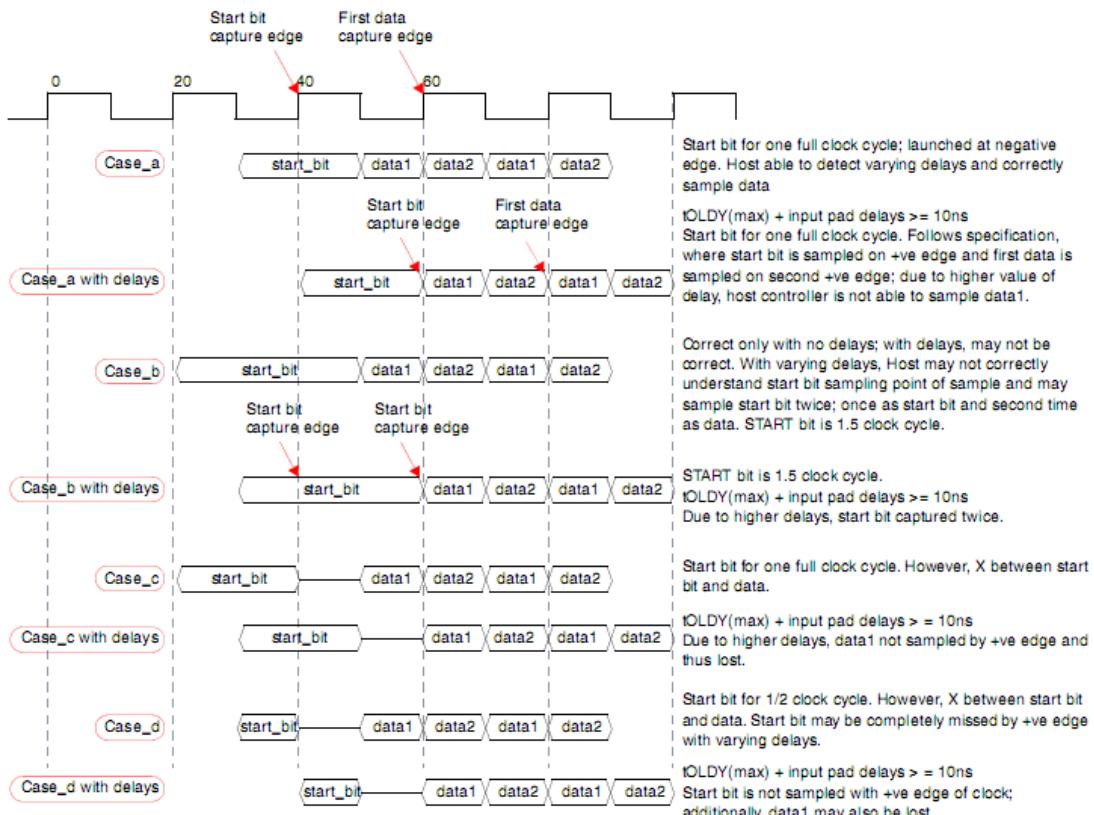


Fig. 14-17 CASES for eMMC 4.5 START bit

#### 4. Reset Command/Moving from DDR50 to SDR12

To reset the mode of operation from DDR50 to SDR12, the following sequence of operations has to be done by the application:

- 1) Issue CMD0.

When CMD0 is received, the card changes from DDR50 to SDR12.

- 2) Program the CLKDIV register with an appropriate value.
- 3) Set DDR\_REG to 0.

*Note: The VOLT\_REG register should not be programmed to 0 while switching from DDR50 to SDR12, since the card is still operating in 1.8V mode after receiving CMD0.*

#### 14.6.8 H/W Reset Operation

When the RST\_n signal goes low, the card enters a pre-idle state from any state other than the inactive state.

## H/W Reset Programming Sequence

The following outlines the steps for the H/W reset programming sequence:

- 1) Program CMD12 to end any transfer in process.
- 2) Wait for DTO, even if no response is sent back by the card.
- 3) Set the following resets:
  - DMA reset– CTRL[2]
  - FIFO reset – CTRL[1] bits

*Note: The above steps are required only if a transfer is in process.*

- 4) Program the CARD\_RESET register with a value of 0; this can be done at any time when the card is connected to the controller. This programming asserts the RST\_n signal and resets the card.
- 5) Wait for minimum of 1  $\mu$ s or cclk\_in period, whichever is greater
- 6) After a minimum of 1  $\mu$ s, the application should program a value of 0 into the CARD\_RESET register. This de-asserts the RST\_n signal and takes the card out of reset.
- 7) The application can program a new CMD only after a minimum of 200  $\mu$ s after the de-assertion of the RST\_n signal, as per the MMC 4.41 standard.

*Note: For backward compatibility, the RST\_n signal is temporarily disabled in the card by default. The host may need to set the signal as either permanently enabled or permanently disabled before it uses the card.*

### 14.6.9 FBE Scenarios

An FBE occurs due to an AHB error response on the AHB bus. This is a system error, so the software driver should not perform any further programming to the Host. The only recovery mechanism from such scenarios is to do one of the following:

- Issue a hard reset by asserting the reset\_n signal
- Do a program controller reset by writing to the CTRL[0] register

### FIFO Overflow and Underflow

During normal data transfer conditions, FIFO overflow and underflow will not occur. However if there is a programming error, then FIFO overflow/underflow can result. For example, consider the following scenarios.

- For transmit: PBL=4, Tx watermark = 1. For the above programming values, if the FIFO has only one location empty, it issues a dma\_req to IDMAC FSM. Due to PBL value=4, the IDMAC FSM performs 4 pushes into the FIFO. This will result in a FIFO overflow interrupt.
- For receive: PBL=4, Rx watermark = 1. For the above programming values, if the FIFO has only one location filled, it issues a dma\_req to IDMAC FSM. Due to PBL value=4, the IDMAC FSM performs 4 pops to the FIFO. This will result in a FIFO underflow interrupt.

The driver should ensure that the number of bytes to be transferred as indicated in the descriptor should be a multiple of 4bytes with respect to H\_DATA\_WIDTH=32. For example, if the BYTCNT = 13, the number of bytes indicated in the descriptor should be 16 for H\_DATA\_WIDTH=32.

### Programming of PBL and Watermark Levels

The DMAC performs data transfers depending on the programmed PBL and threshold values.

Table 14-16 PBL and Watermark Levels

PBL (Number of transfers)	Tx/Rx Watermark Value
1	greater than or equal to 1
4	greater than or equal to 4
8	greater than or equal to 8
16	greater than or equal to 16
32	greater than or equal to 32
64	greater than or equal to 64
128	greater than or equal to 128
256	greater than or equal to 256

#### 14.6.10 Variable Delay/Clock Generation

Variable delay mechanism for the cclk\_in\_drv is optional, but it can be useful in order to meet a range of hold-time requirements across modes. Variable delay mechanism for the cclk\_in\_sample is mandatory and is required to achieve the correct sampling point for data.

cclk\_in/cclk\_in\_sample/ cclk\_in\_drv is generated by Clock Generation Unit (CLKGEN) with variable delay mechanism, which includes Phase Shift Unit and Delay Line Unit selectable.

The Phase Shift Unit can shift cclk\_in\_sample/cclk\_in\_drv by 0/90/180/270-degree relative to cclk\_in, controlled by *sample\_degree/drv\_degree*.

The Delay Line Unit can shift cclk\_in\_sample/cclk\_in\_drv in the unit of 40ps~80ps for every delay element. The delay unit number is determined by *sample\_delaynum/drv\_delaynum*, and enabled by *sample\_sel/drv\_sel*.

cclk\_in is generated by cclkin divided by 2. cclk\_in\_drv and cclk\_in\_sample clocks are phase-shifted with delayed versions of cclk\_in. All clocks are recommended to have a 50% duty cycle; DDR modes must have 50% duty cycles.

The architecture is as follows.

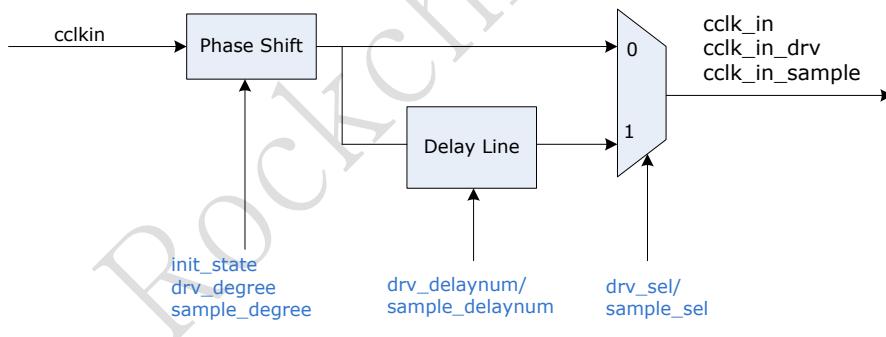


Fig. 14-18 Clock Generation Unit

The control signals for different Host Controller instance are shown as follows:

Table 14-17 Configuration for SDMMC Clock Generation

Signal Name	Source	Default	Description
init_state	CRU_SDMMC_CON0[0]	0	Soft initial state for phase shift.
drv_degree [1:0]	CRU_SDMMC_CON0[2:1]	2	Phase shift for cclk_in_drv. 0: 0-degree 1: 90-degree 2: 180-degree

			3: 270-degree
drv_delaynum [7:0]	CRU_SDMMC_CON0[10:3]	0	Element number in delay line for cclk_in_drv
drv_sel	CRU_SDMMC_CON0[11]	0	cclk_in_drv source selection: 0: use clock after phase_shift 1: use clock after phase_shift and delay line
sample_degree [1:0]	CRU_SDMMC_CON1[1:0]	0	Phase shift for cclk_in_sample. 0: 0-degree 1: 90-degree 2: 180-degree 3: 270-degree
sample_delaynum [7:0]	CRU_SDMMC_CON1[9:2]	0	Element number in delay line for cclk_in_sample
sample_sel	CRU_SDMMC_CON1[10]	0	cclk_in_sample source selection: 0: use clock after phase_shift 1: use clock after phase_shift and delay line

Table 14-18 Configuration for SDIO0 Clock Generation

Signal Name	Source	Default	Description
init_state	CRU_SDIO0_CON0[0]	0	Soft initial state for phase shift.
drv_degree [1:0]	CRU_SDIO0_CON0[2:1]	2	Phase shift for cclk_in_drv. 0: 0-degree 1: 90-degree 2: 180-degree 3: 270-degree
drv_delaynum [7:0]	CRU_SDIO0_CON0[10:3]	0	Element number in delay line for cclk_in_drv
drv_sel	CRU_SDIO0_CON0[11]	0	cclk_in_drv source selection: 0: use clock after phase_shift 1: use clock after phase_shift and delay line
sample_degree [1:0]	CRU_SDIO0_CON1[1:0]	0	Phase shift for cclk_in_sample. 0: 0-degree 1: 90-degree 2: 180-degree 3: 270-degree
sample_delaynum [7:0]	CRU_SDIO0_CON1[9:2]	0	Element number in delay line for cclk_in_sample
sample_sel	CRU_SDIO0_CON1[10]	0	cclk_in_sample source selection: 0: use clock after phase_shift

			1: use clock after phase_shift and delay line
--	--	--	--

Table 14-19 Configuration for SDIO1 Clock Generation

Signal Name	Source	Default	Description
init_state	CRU_SDIO1_CON0[0]	0	Soft initial state for phase shift.
drv_degree [1:0]	CRU_SDIO1_CON0[2:1]	2	Phase shift for cclk_in_drv. 0: 0-degree 1: 90-degree 2: 180-degree 3: 270-degree
drv_delaynum [7:0]	CRU_SDIO1_CON0[10:3]	0	Element number in delay line for cclk_in_drv
drv_sel	CRU_SDIO1_CON0[11]	0	cclk_in_drv source selection: 0: use clock after phase_shift 1: use clock after phase_shift and delay line
sample_degree [1:0]	CRU_SDIO1_CON1[1:0]	0	Phase shift for cclk_in_sample. 0: 0-degree 1: 90-degree 2: 180-degree 3: 270-degree
sample_delaynum [7:0]	CRU_SDIO1_CON1[9:2]	0	Element number in delay line for cclk_in_sample
sample_sel	CRU_SDIO1_CON1[10]	0	cclk_in_sample source selection: 0: use clock after phase_shift 1: use clock after phase_shift and delay line

Table 14-20 Configuration for eMMC Clock Generation

Signal Name	Source	Default	Description
init_state	CRU_EMMC_CON0[0]	0	Soft initial state for phase shift.
drv_degree [1:0]	CRU_EMMC_CON0[2:1]	2	Phase shift for cclk_in_drv. 0: 0-degree 1: 90-degree 2: 180-degree 3: 270-degree
drv_delaynum [7:0]	CRU_EMMC_CON0[10:3]	0	Element number in delay line for cclk_in_drv
drv_sel	CRU_EMMC_CON0[11]	0	cclk_in_drv source selection: 0: use clock after phase_shift 1: use clock after phase_shift and delay line

sample_degree [1:0]	CRU_EMMC_CON1[1:0]	0	Phase shift for cclk_in_sample. 0: 0-degree 1: 90-degree 2: 180-degree 3: 270-degree
sample_delaynum [7:0]	CRU_EMMC_CON1[9:2]	0	Element number in delay line for cclk_in_sample
sample_sel	CRU_EMMC_CON1[10]	0	cclk_in_sample source selection: 0: use clock after phase_shift 1: use clock after phase_shift and delay line

The following outlines the steps for clock generation sequence:

- 1) Assert init\_state to soft reset the CLKGEN.
- 2) Configure drv\_degree/sample\_degree.
- 3) If fine adjustment required, delay line can be used by configuring drv\_delaynum/sample\_delaynum and drv\_sel/sample\_sel.
- 4) Dis-assert init\_state to start CLKGEN.

#### 14.6.11 Variable Delay Tuning

Tuning is defined by SD and MMC cards to determine the correct sampling point required for the host, especially for the speed modes SDR104 and HS200 where the output delays from the cards can be up to 2 UI. Tuning is required for other speed modes—such as DDR50—even though the output delay from the card is less than one cycle.

Command for tuning is different for different cards.

- SD Memory Card:
  - CMD19 – SD card for SDR50 and SDR104 speed modes. Tuning data is defined by card specifications.
  - CMD6 – SD card for speed modes not supporting CMD19. Tuning data is the 64byte SD status.
- Multimedia Card:
  - CMD21 – MMC card for HS200 speed mode. Tuning data is defined by card specifications.
  - CMD8 – MMC card for speed modes not supporting CMD21. Tuning data is 512 byte ExtCSD data.

The following is the procedure for variable delay tuning:

- 1) Set a phase shift of 0-degree on cclk\_in\_sample.
- 2) Send the Tuning command to the card; the card in turn sends an R1 response on the CMD line and tuning data on the DAT line.
- 3) If the host sees any of the errors—start bit error, data crc error, end bit error, data read time-out, response crc error, response error—then the sampling point is incorrect.
- 4) Send CMD12 to bring the host controller state machines to idle.
  - The card may treat CMD12 as an invalid command because the card has successfully sent the tuning data,

and it cannot send a response.

- The host controller may generate a response time-out interrupt that must be cleared by software.
- 5) Repeat steps 2) to 4) by increasing the phase shift value or delay element number on cclk\_in\_sample until the correct sampling point is received such that the host does not see any of the errors.
  - 6) Mark this phase shift value as the starting point of the sampling window.
  - 7) Repeat steps 2 to 4 by increasing the phase shift value or delay element number on cclk\_in\_sample until the host sees the errors starting to come again or the phase shift value reaches 360-degree.
  - 8) Mark the last successful phase shift value as the ending point of the sampling window.

A window is established where the tuning block is matched. For example, for a scenario where the tuning block is received correctly for a phase shift window of 90-degree and 180-degree, then an appropriate sampling point is established as 135-degree. Once a sampling point is established, no errors should be visible in the tuning block.

#### **14.6.12 Package Command**

In order to reduce overhead, read and write commands can be packed in groups of commands-either all read or all write-that transfer the data for all commands in the group in one transfer on the bus.

Packed commands can be of two types:

- Packed Write: CMD23 → CMD25
- Packed Read: CMD23 → CMD25 → CMD23 → CMD18

Packed commands are put in packets by the application software and are transparent to the core. For more information on packed commands, refer to the eMMC specification.

#### **14.6.13 Card Detection Method**

There are many methods for SDMMC/SDIO0/SDIO1 card detection.

(1) Method1: Using CDETECT register, which is value on card\_detect\_n input port. 0 represents presence of card. Commonly for SDMMC/SDIO0/SDIO1.

(2) Method2: Using card detection unit, outputting host interrupt (*IRQ\_ID[64]/[65]/[66]*). The card detection unit looks for any changes in the card-detect signals for card insertion or card removal. It filters out the debounces associated with mechanical insertion or removal, and generates one interrupt to the host. You can program the debounce filter value in DEBNCE[23:0]. Following figure illustrates the timing for card-detect signals. Commonly for SDMMC/SDIO0/SDIO1.

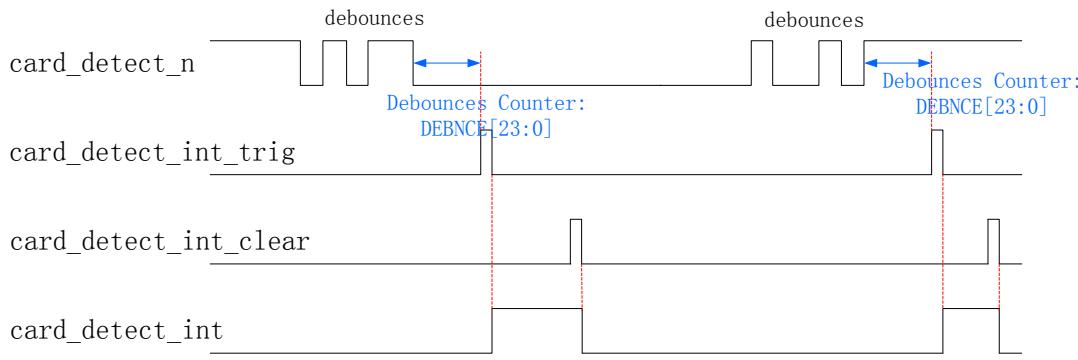


Fig. 14-19 Card Detection Method 2

- Method3: Using card detection unit in GRF, outputting *sdmmc\_detect\_dual\_edge\_int(IRQ\_ID[138])*, only available for SDMMC. Similar to Method2, except that the debounce is selecting from 5ms/15ms/35ms/50ms; and the insertion/removal detection interrupt can be enabled or cleared respectively. The detailed register information is:

Table 14-21 Register for SDMMC Card Detection Method 3

Signal Name	Source	Default	Description
sd_detectn_rise_edge_irq_en	GRF_SOC_CON11[0]	0	sdmmc detect_n signal rise edge interrupt enable. 1'b1: enable 1'b0: disable
sd_detectn_rise_edge_irq_pd	GRF_SOC_CON11[1]	0	sdmmc detect_n rise edge interrupt pending status. Write 1 to clear the status.
sd_detectn_fall_edge_irq_en	GRF_SOC_CON11[2]	0	sdmmc detect_n signal fall edge interrupt enable. 1'b1: enable 1'b0: disable
sd_detectn_fall_edge_irq_pd	GRF_SOC_CON11[3]	0	sdmmc detect_n fall edge interrupt pending status. Write 1 to clear the status.
grf_filter_cnt_sel	GRF_SOC_CON12[1:0]	0	the counter select for sd card detect filter: 2'b00: 5ms 2'b01: 15ms 2'b10: 35ms 2'b11: 50ms

- Method4: Using **card\_detect\_n** for interrupt source, connecting to *IRQ\_ID[131]/[132]/[133]* directly. Commonly for SDMMC/SDIO0/SDIO1.

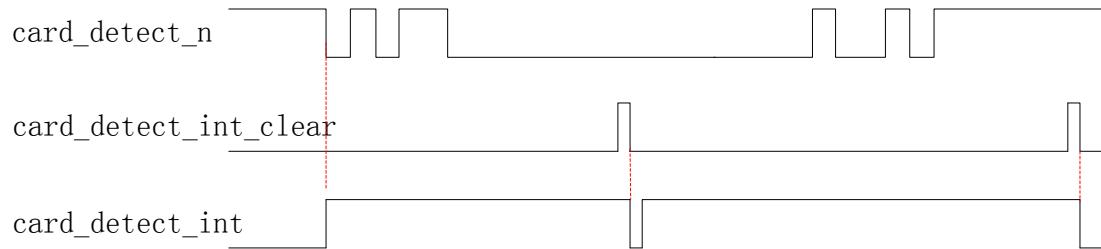


Fig. 14-20 Card Detection Method 4

## Chapter 15 Embedded SRAM

### 15.1 Overview

The Embedded SRAM is the AXI slave device, which supports read and write access to provide system fast access data storage.

#### 15.1.1 Features supported

- Provide 96KB access space
- Support security and non-security access
- Security or non-security space is software programmable
- Security space is nx4KB(up to whole memory space)
- Support 64bit AXI bus

#### 15.1.2 Features not supported

- Don't support AXI lock transaction
- Don't support AXI exclusive transaction
- Don't support AXI cache function
- Don't support AXI protection function

### 15.2 Block Diagram

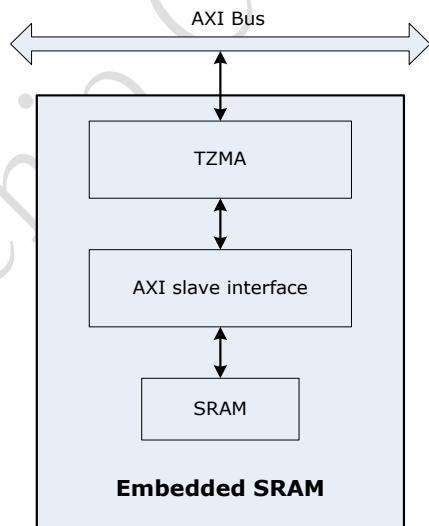


Fig. 15-1 Embedded SRAM block diagram

### 15.3 Function Description

#### 15.3.1 TZMA

Please refer to 5.3.3 for TZMA functional description.

### **15.3.2 AXI slave interface**

The AXI slave interface is bridge which translate AXI bus access to SRAM interface.

### **15.3.3 Embedded SRAM access path**

The Embedded SRAM can only be accessed by Cortex-A17 and DMAC\_BUS.

### **15.3.4 Remap**

The Embedded SRAM support remap.

Before remap, the Embedded SRAM address range is 0xff70\_0000~0xff71\_7fff,

After set remap, (ref Security GRF register SGRF\_SCON0, bit[7]), the system can still access the Embedded SRAM by the old address. at same time, the system also can access the Embedded SRAM by the new address 0xffffe\_0000 ~ 0xfffff\_7fff (include the bootaddr)

## Chapter 16 I2S/PCM Controller (8 channel)

### 16.1 Overview

The I2S/PCM controller is designed for interfacing between the AHB bus and the I2S bus.

The I2S bus (Inter-IC sound bus) is a serial link for digital audio data transfer between devices in the system and was invented by Philips Semiconductor. Now it is widely used by many semiconductor manufacturers.

Devices often use the I2S bus are ADC, DAC, DSP, CPU, etc. With the I2S interface, we can connect audio devices and the embedded SoC platform together and provide an audio interface solution for the system.

Not only I2S but also PCM mode surround audio output (up to 7.1channel) and stereo input are supported in I2S/PCM controller.

- Support five internal 32-bit wide and 32-location deep FIFOs, four for transmitting and one for receiving audio data
- Support AHB bus interface
- Support 16 ~ 32 bits audio data transfer
- Support master and slave mode
- Support DMA handshake interface and configurable DMA water level
- Support transmit FIFO empty, underflow, receive FIFO full, overflow interrupt and all interrupts can be masked
- Support configurable water level of transmit FIFO empty and receive FIFO full interrupt
- Support combine interrupt output
- Support 2,4,6,8 channels audio transmitting in I2S and PCM mode
- Support 2 channels audio receiving in I2S and PCM mode
- Support up to 192kHz sample rate
- Support I2S normal, left and right justified mode serial audio data transfer
- Support PCM early, late1, late2, late3 mode serial audio data transfer
- Support MSB or LSB first serial audio data transfer
- Support 16 to 31 bit audio data left or right justified in 32-bit wide FIFO
- Support two 16-bit audio data store together in one 32-bit wide location
- Support 3 independent LRCK signals, one for receiving and two for transmitting audio data
- Support configurable SCLK and LRCK polarity
- Support SCLK is equivalent to MCLK divided by an even number range from 2 to 64 in master mode

## 16.2 Block Diagram

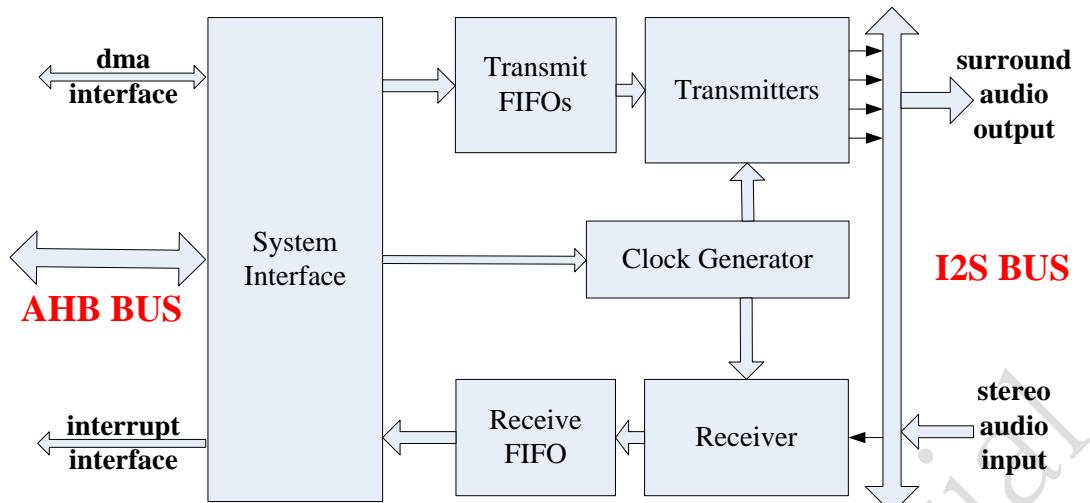


Fig. 16-1 I2S/PCM controller (8 channel) Block Diagram

### System Interface

The system interface implements the AHB slave operation. It contains not only control registers of transmitters and receiver inside but also interrupt and DMA handshake interface.

### Clock Generator

The Clock Generator implements clock generation function. The input source clock to the module is MCLK\_I2S, and by the divider of the module, the clock generator generates SCLK and LRCK to transmitter and receiver.

### Transmitters

The Transmitters implement transmission operation. The transmitters can act as either master or slave, with I2S or PCM mode surround (up to 7.1 channel) serial audio interface.

### Receiver

The Receiver implements receive operation. The receiver can act as either master or slave, with I2S or PCM mode stereo serial audio interface.

### Transmit FIFOs

The Transmit FIFOs are the buffer to store transmitted audio data. Each of the size of the four FIFOs is 32bits x 32.

### Receive FIFO

The Receive FIFO is the buffer to store received audio data. The size of the FIFO is 32bits x 32.

## 16.3 Function description

In the I2S/PCM controller, there are four conditions: transmitter-master & receiver-master; transmitter-master & receiver-slave; transmitter-slave & receiver-master; transmitter-slave & receiver-slave.

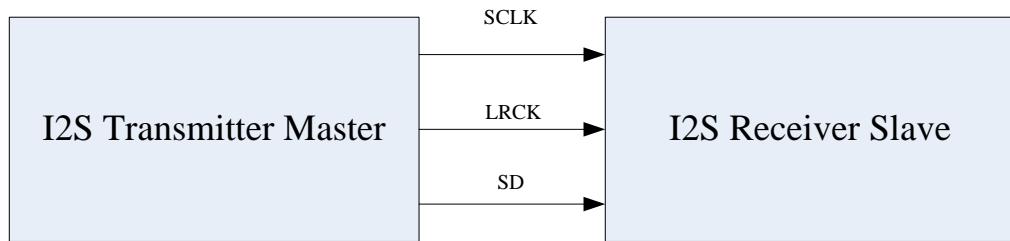


Fig. 16-2 I2S transmitter-master & receiver-slave condition

When transmitter acts as a master, it sends all signals to receiver (slave), and CPU control when to send clock and data to the receiver. When acting as a slave, SD signal still goes from transmitter to receiver, but SCLK and LRCK signals are from receiver (master) to transmitter. Based on three interface specifications, transmitting data should be ready before transmitter receives SCLK and LRCK signals. CPU should know when the receiver to initialize a transaction and when to send data.

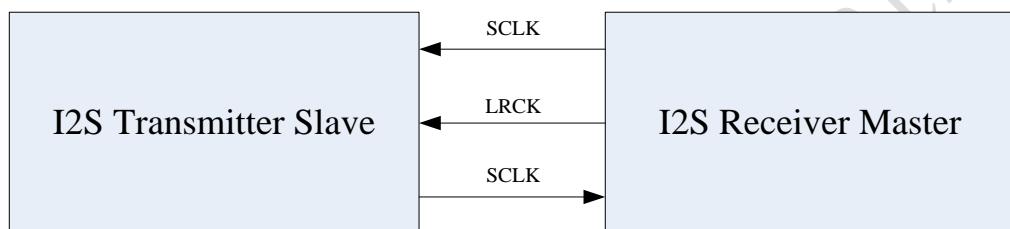


Fig. 16-3 I2S transmitter-slave& receiver-master condition

When the receiver acts as a master, it sends SCLK and LRCK signals to the transmitter (slave) and receives serial data. So CPU must tell the transmitter when to start a transaction for it to prepare transmitting data then start a transfer and send clock and channel-select signals. When the receiver acts as a slave, CPU should only do initial setting and wait for all signals and then start reading data.

Before transmitting or receiving data, CPU need do initial setting to the I2S register. These includes CPU settings, I2S interface registers settings, and maybe the embedded SoC platform settings. These registers must be set before starting data transfer.

### 16.3.1 i2s normal mode

This is the waveform of I2S normal mode. For LRCK (i2s\_lrck\_rx/i2s\_lrck\_tx0) signal, it goes low to indicate left channel and high to right channel. For SD (i2s\_sdo0, i2s\_sdo1, i2s\_sdo2, i2s\_sdo3, i2s\_sdi) signal, it transfers MSB or LSB first and sends the first bit one SCLK clock cycle after LRCK changes. The range of SD signal width is from 16 to 32bits.

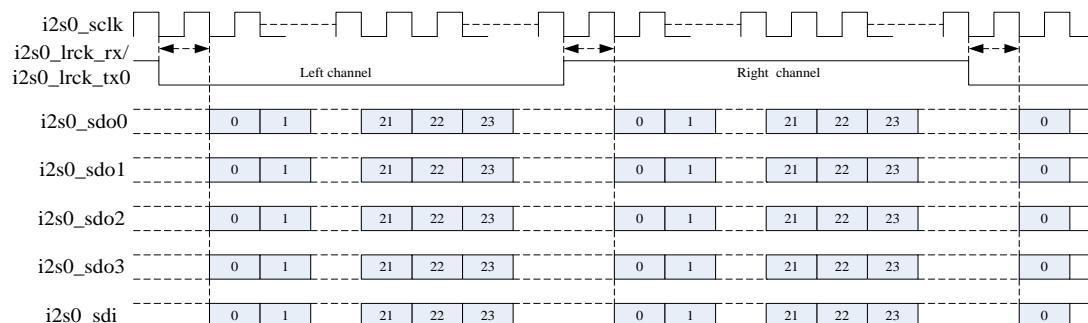


Fig. 16-4 I2S normal mode timing format

### 16.3.2 i2s left justified mode

This is the waveform of I2S left justified mode. For LRCK (i2s\_lrck\_rx / i2s\_lrck\_tx0) signal, it goes high to indicate left channel and low to right channel. For SD (i2s\_sdo0, i2s\_sdo1, i2s\_sdo2, i2s\_sdo3, i2s\_sdi) signal, it transfers MSB or LSB first and sends the first bit at the same time when LRCK changes. The range of SD signal width is from 16 to 32bits.

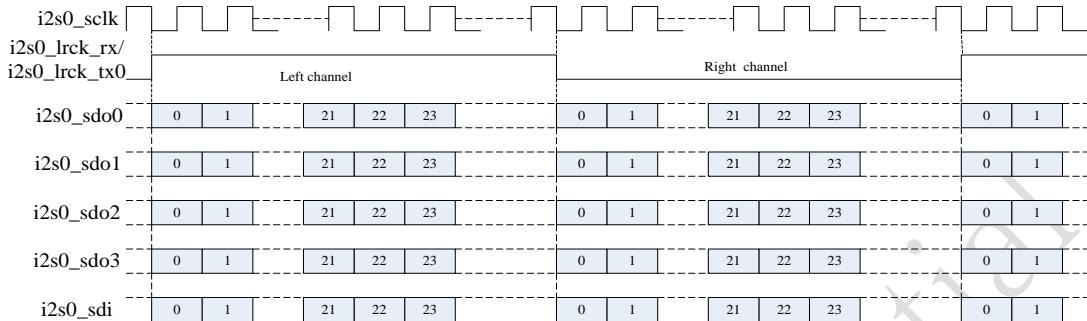


Fig. 16-5 I2S left justified mode timing format

### 16.3.3 i2s right justified mode

This is the waveform of I2S right justified mode. For LRCK (i2s\_lrck\_rx / i2s\_lrck\_tx0) signal, it goes high to indicate left channel and low to right channel. For SD (i2s\_sdo0, i2s\_sdo1, i2s\_sdo2, i2s\_sdo3, i2s\_sdi) signal, it transfers MSB or LSB first; but different from I2S normal or left justified mode, its data is aligned to last bit at the edge of the LRCK signal. The range of SD signal width is from 16 to 32bits.

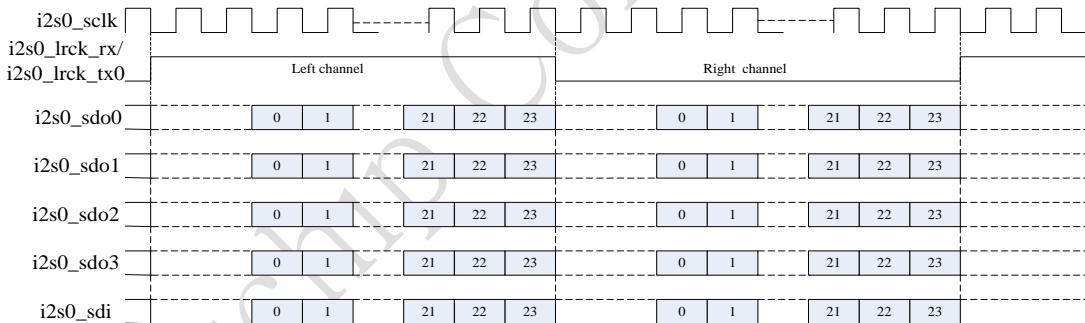


Fig. 16-6 I2S right justified mode timing format

### 16.3.4 PCM early mode

This is the waveform of PCM early mode. For LRCK (i2s\_lrck\_rx / i2s\_lrck\_tx0) signal, it goes high to indicate the start of a group of audio channels. For SD (i2s\_sdo0, i2s\_sdi) signal, it transfers MSB or LSB first and sends the first bit at the same time when LRCK goes high. The range of SD signal width is from 16 to 32bits.

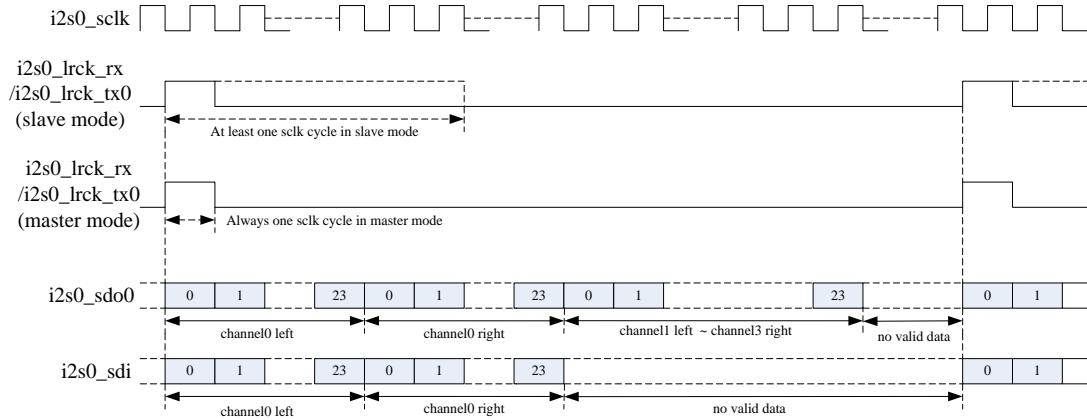


Fig. 16-7 PCM early modetiming format

### 16.3.5 PCM late1 mode

This is the waveform of PCM early mode. For LRCK (i2s\_lrck\_rx / i2s\_lrck\_tx0) signal, it goes high to indicate the start of a group of audio channels. For SD (i2s\_sdo0, i2s\_sdi) signal, it transfers MSB or LSB first and sends the first bit one SCLK clock cycle after LRCK goes high. The range of SD signal width is from 16 to 32bits.

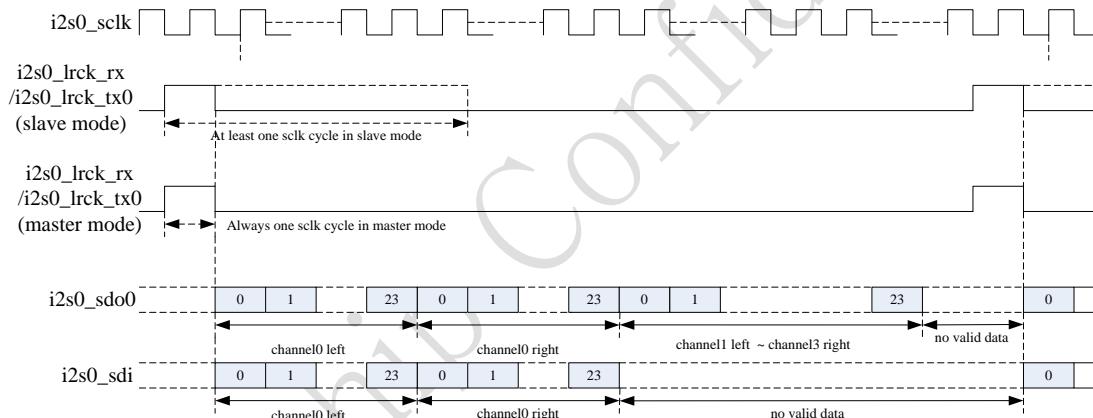


Fig. 16-8 PCM late1 modetiming format

### 16.3.6 PCM late2 mode

This is the waveform of PCM early mode. For LRCK (i2s\_lrck\_rx / i2s\_lrck\_tx0) signal, it goes high to indicate the start of a group of audio channels. For SD (i2s\_sdo0, i2s\_sdi) signal, it transfers MSB or LSB first and sends the first bit two SCLK clock cycles after LRCK goes high. The range of SD signal width is from 16 to 32bits.

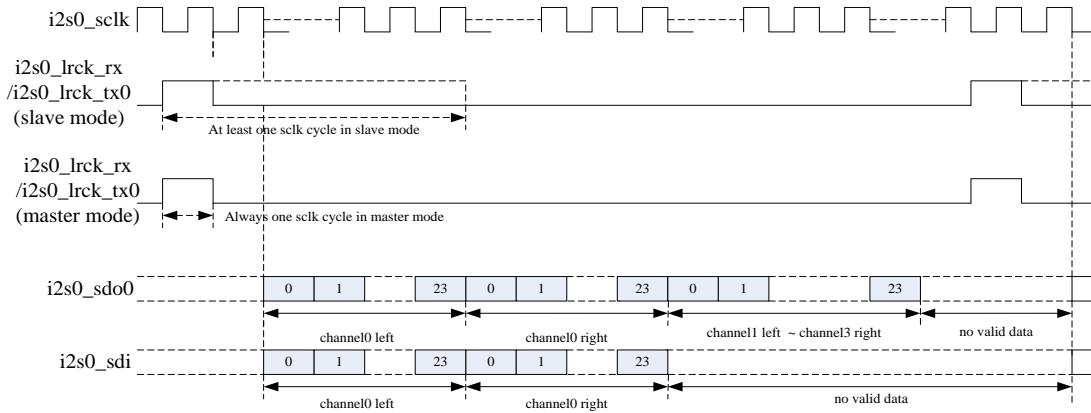


Fig. 16-9 PCM late2 modetiming format

### 16.3.7 PCM late3 mode

This is the waveform of PCM early mode. For LRCK (i2s\_lrck\_rx / i2s\_lrck\_tx0) signal, it goes high to indicate the start of a group of audio channels. For SD (i2s\_sdo0, i2s\_sdi) signal, it transfers MSB or LSB first and sends the first bit three SCLK clock cycles after LRCK goes high. The range of SD signal width is from 16 to 32bits.

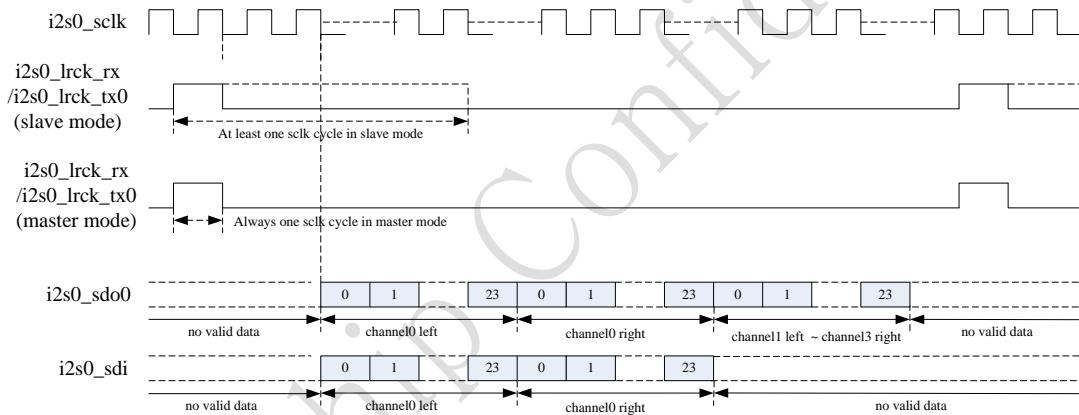


Fig. 16-10 PCM late3 modetiming format

## 16.4 Register Description

This section describes the control/status registers of the design.

### 16.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
I2S_TXCR	0x0000	W	0x0000000f	transmit operation control register
I2S_RXCR	0x0004	W	0x0000000f	receive operation control register
I2S_CKR	0x0008	W	0x00071f1f	clock generation register
I2S_FIFOLR	0x000c	W	0x00000000	FIFO level register
I2S_DMACR	0x0010	W	0x001f0000	DMA control register
I2S_INTCR	0x0014	W	0x01f00000	interrupt control register
I2S_INTSR	0x0018	W	0x00000000	interrupt status register
I2S_XFER	0x001c	W	0x00000000	Transfer Start Register
I2S_CLR	0x0020	W	0x00000000	SCLK domain logic clear Register
I2S_TXDR	0x0024	W	0x00000000	Transimt FIFO Data Register

Name	Offset	Size	Reset Value	Description
I2S_RXDR	0x0028	W	0x00000000	Receive FIFO Data Register

Notes:Size: **B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**-WORD (32 bits) access

### 16.4.2 Detail Register Description

#### I2S\_TXCR

Address: Operational Base + offset (0x0000)

transmit operation control register

Bit	Attr	Reset Value	Description
31:23	RO	0x0	reserved
22:17	RW	0x00	<p>RCNT right jusitified counter (Can be written only when XFER[0] bit is 0.) Only vailid in I2S Right justified format and slave tx mode is selected. Start to tramsmit data RCNT sclk cycles after left channel valid.</p>
16:15	RW	0x0	<p>CSR Channel select register (Can be written only when XFER[0] bit is 0.) 0:channel 0 enable 1:channel 0 &amp; channel 1 enable 2:channel 0 &amp; channel 1 &amp; channel 2 enable 3:channel 0 &amp; channel 1 &amp; channel 2 &amp; channel 3 enable</p>
14	RW	0x0	<p>HWT Halfword word transform (Can be written only when XFER[0] bit is 0.) Only valid when VDW select 16bit data. 0:32 bit data valid from AHB/APB bus. Low 16 bit for left channel and high 16 bit for right channel. 1:low 16bit data valid from AHB/APB bus, high 16 bit data invalid.</p>
13	RO	0x0	reserved
12	RW	0x0	<p>SJM Store justified mode (Can be written only when XFER[0] bit is 0.) 16bit~31bit DATA stored in 32 bits width fifo. If VDW select 16bit data, this bit is valid only when HWT select 0.Because if HWT is 1, every fifo unit contain two 16bit data and 32 bit space is full, it is impossible to choose justified mode. 0:right justified 1:left justified</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x0	FBM First Bit Mode (Can be written only when XFER[0] bit is 0.) 0:MSB 1:LSB
10:9	RW	0x0	IBM I2S bus mode (Can be written only when XFER[0] bit is 0.) 0:I2S normal 1:I2S Left justified 2:I2S Right justified 3:reserved
8:7	RW	0x0	PBM PCM bus mode (Can be written only when XFER[0] bit is 0.) 0:PCM no delay mode 1:PCM delay 1 mode 2:PCM delay 2 mode 3:PCM delay 3 mode
6	RO	0x0	reserved
5	RW	0x0	TFS Transfer format select (Can be written only when XFER[0] bit is 0.) 0: I2S format 1: PCM format
4:0	RW	0x0f	VDW Valid Data width (Can be written only when XFER[0] bit is 0.) 0~14:reserved 15:16bit 16:17bit 17:18bit 18:19bit ..... 28:29bit 29:30bit 30:31bit 31:32bit

**I2S\_RXCR**

Address: Operational Base + offset (0x0004)  
receive operation control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14	RW	0x0	HWT Halfword word transform (Can be written only when XFER[1] bit is 0.) Only valid when VDW select 16bit data. 0:32 bit data valid to AHB/APB bus. Low 16 bit for left channel and high 16 bit for right channel. 1:low 16bit data valid to AHB/APB bus, high 16 bit data invalid.
13	RO	0x0	reserved
12	RW	0x0	SJM Store justified mode (Can be written only when XFER[1] bit is 0.) 16bit~31bit DATA stored in 32 bits width fifo. If VDW select 16bit data, this bit is valid only when HWT select 0.Because if HWT is 1, every fifo unit contain two 16bit data and 32 bit space is full, it is impossible to choose justified mode. 0:right justified 1:left justified
11	RW	0x0	FBM First Bit Mode (Can be written only when XFER[1] bit is 0.) 0:MSB 1:LSB
10:9	RW	0x0	IBM I2S bus mode (Can be written only when XFER[1] bit is 0.) 0:I2S normal 1:I2S Left justified 2:I2S Right justified 3:reserved
8:7	RW	0x0	PBM PCM bus mode (Can be written only when XFER[1] bit is 0.) 0:PCM no delay mode 1:PCM delay 1 mode 2:PCM delay 2 mode 3:PCM delay 3 mode
6	RO	0x0	reserved
5	RW	0x0	TFS Transfer format select (Can be written only when XFER[1] bit is 0.) 0:i2s 1:pcm

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:0	RW	0x0f	VDW Valid Data width (Can be written only when XFER[1] bit is 0.) 0~14:reserved 15:16bit 16:17bit 17:18bit 18:19bit ..... 28:29bit 29:30bit 30:31bit 31:32bit

**I2S\_CKR**

Address: Operational Base + offset (0x0008)  
 clock generation register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27	RW	0x0	MSS Master/slave mode select (Can be written only when XFER[1] or XFER[0] bit is 0.) 0:master mode(sclk output) 1:slave mode(sclk input)
26	RW	0x0	CKP Sclk polarity (Can be written only when XFER[1] or XFER[0] bit is 0.) 0: sample data at posedge sclk and drive data at negedge sclk 1: sample data at negedge sclk and drive data at posedge sclk

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
25	RW	0x0	<p>RLP Receive lrck polarity (Can be written only when XFER[1] or XFER[0] bit is 0.)</p> <p>0: normal polarity (I2S normal: low for left channel, high for right channel I2S left/right just: high for left channel, low for right channel PCM start signal: high valid)</p> <p>1: opposite polarity (I2S normal: high for left channel, low for right channel I2S left/right just: low for left channel, high for right channel PCM start signal: low valid)</p>
24	RW	0x0	<p>TLP Transmit lrck polarity (Can be written only when XFER[1] or XFER[0] bit is 0.)</p> <p>0: normal polarity (I2S normal: low for left channel, high for right channel I2S left/right just: high for left channel, low for right channel PCM start signal: high valid)</p> <p>1: opposite polarity (I2S normal: high for left channel, low for right channel I2S left/right just: low for left channel, high for right channel PCM start signal: low valid)</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23:16	RW	0x07	<p>MDIV mclk divider (Can be written only when XFER[1] or XFER[0] bit is 0.) Serial Clock Divider = Fmclk / Ftxsclk-1.(mclk frequency / txsclk frequency-1)</p> <p>0 :Fmclk=Ftxsclk; 1 :Fmclk=2*Ftxsclk; 2,3 :Fmclk=4*Ftxsclk; 4,5 :Fmclk=6*Ftxsclk; ..... 60,61:Fmclk=62*Ftxsclk; 62,63:Fmclk=64*Ftxsclk; ..... 252,253:Fmclk=254*Ftxsclk; 254,255:Fmclk=256*Ftxsclk;</p>
15:8	RW	0x1f	<p>RSD Receive sclk divider (Can be written only when XFER[1] or XFER[0] bit is 0.) Receive sclk divider= Fsclk/Frxlrck</p> <p>0~30:reserved 31: 32fs 32: 33fs 33: 34fs 34: 35fs ..... 253: 254fs 254: 255fs 255: 256fs</p>
7:0	RW	0x1f	<p>TSD Transmit sclk divider (Can be written only when XFER[1] or XFER[0] bit is 0.) Transmit sclk divider=Ftxsclk/Ftxlrck</p> <p>0~30:reserved 31: 32fs 32: 33fs 33: 34fs 34: 35fs ..... 253: 254fs 254: 255fs 255: 256fs</p>

**I2S\_FIFOLR**

Address: Operational Base + offset (0x000c)

FIFO level register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved
29:24	RO	0x00	RFL Receive FIFO Level Contains the number of valid data entries in the receive FIFO.
23:18	RO	0x00	TFL3 Transmit FIFO3 Level Contains the number of valid data entries in the transmit FIFO3.
17:12	RO	0x00	TFL2 Transmit FIFO2 Level Contains the number of valid data entries in the transmit FIFO2.
11:6	RO	0x00	TFL1 Transmit FIFO1 Level Contains the number of valid data entries in the transmit FIFO1.
5:0	RO	0x00	TFL0 Transmit FIFO0 Level Contains the number of valid data entries in the transmit FIFO0.

**I2S\_DMACR**

Address: Operational Base + offset (0x0010)

DMA control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	RDE Receive DMA Enable 0 : Receive DMA disabled 1 : Receive DMA enabled
23:21	RO	0x0	reserved
20:16	RW	0x1f	RDL Receive Data Level This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1.
15:9	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	RW	0x0	TDE Transmit DMA Enable 0 : Transmit DMA disabled 1 : Transmit DMA enabled
7:5	RO	0x0	reserved
4:0	RW	0x00	TDL Transmit Data Level This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the TXFIFO(TXFIFO0 if CSR=00;TXFIFO1 if CSR=01,TXFIFO2 if CSR=10,TXFIFO3 if CSR=11)is equal to or below this field value.

**I2S\_INTCR**

Address: Operational Base + offset (0x0014)  
interrupt control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:20	RW	0x1f	RFT Receive FIFO Threshold When the number of receive FIFO entries is more than or equal to this threshold plus 1, the receive FIFO full interrupt is triggered.
19	RO	0x0	reserved
18	WO	0x0	RXOIC RX overrun interrupt clear Write 1 to clear RX overrun interrupt.
17	RW	0x0	RXOIE RX overrun interrupt enable 0:disable 1:enable
16	RW	0x0	RXFIE RX full interrupt enable 0:disable 1:enable
15:9	RO	0x0	reserved
8:4	RW	0x00	TFT Transmit FIFO Threshold When the number of transmit FIFO (TXFIFO0 if CSR=00; TXFIFO1 if CSR=01, TXFIFO2 if CSR=10, TXFIFO3 if CSR=11) entries is less than or equal to this threshold, the transmit FIFO empty interrupt is triggered.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RO	0x0	reserved
2	WO	0x0	TXUIC TX underrun interrupt clear Write 1 to clear TX underrun interrupt.
1	RW	0x0	TXUIE TX underrun interrupt enable 0:disable 1:enable
0	RW	0x0	TXEIE TX empty interrupt enable 0:disable 1:enable

**I2S\_INTSR**

Address: Operational Base + offset (0x0018)

interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:18	RO	0x0	reserved
17	RO	0x0	RXOI RX overrun interrupt 0:inactive 1:active
16	RO	0x0	RXFI RX full interrupt 0:inactive 1:active
15:2	RO	0x0	reserved
1	RO	0x0	TXUI TX underrun interrupt 0:inactive 1:active
0	RO	0x0	TXEI TX empty interrupt 0:inactive 1:active

**I2S\_XFER**

Address: Operational Base + offset (0x001c)

Transfer Start Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	RXS RX Transfer start bit 0:stop RX transfer. 1:start RX transfer

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	TXS TX Transfer start bit 0:stop TX transfer. 1:start TX transfer

**I2S\_CLR**

Address: Operational Base + offset (0x0020)

SCLK domain logic clear Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	RXC RX logic clear This is a self cleared bit. Write 1 to clear all receive logic.
0	RW	0x0	TXC TX logic clear This is a self cleared bit. Write 1 to clear all transmit logic.

**I2S\_TXDR**

Address: Operational Base + offset (0x0400~0x7FC)

Transmit FIFO Data Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	WO	0x00000000	TXDR Transmit FIFO Data Register When it is written to, data are moved into the transmit FIFO.

**I2S\_RXDR**

Address: Operational Base + offset (0x0800~0xBFC)

Receive FIFO Data Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	RXDR Receive FIFO Data Register When the register is read, data in the receive FIFO is accessed.

**16.5 Interface description**

<b>Module Pin</b>	<b>Direction</b>	<b>Pad Name</b>	<b>IOMUX Setting</b>
i2s_sdi	I	IO_I2Ssdi_AUDI_Ogpio6a3	GRF_GPIO6A_IOMUX[6]=1
i2s_clk	O	IO_I2Sclk_AUDI_Ogpio6b0	GRF_GPIO6B_IOMUX[0]=1
i2s_sclk	I/O	IO_I2Ssclk_AUD_IOgpio6a0	GRF_GPIO6A_IOMUX[0]=1

i2s_lrck_rx	I/O	IO_I2Slrckrx_A UDIOgpio6a1	GRF_GPIO6A_IOMUX[2]=1
i2s_lrck_tx	I/O	IO_I2Slrcktx_A UDIOgpio6a2	GRF_GPIO6A_IOMUX[4]=1
i2s_sdo0	O	IO_I2Ssdo0_AU DIOgpio6a4	GRF_GPIO6A_IOMUX[8]=1
i2s_sdo1	O	IO_I2Ssdo1_AU DIOgpio6a5	GRF_GPIO6A_IOMUX[10]=1
i2s_sdo2	O	IO_I2Ssdo2_AU DIOgpio6a6	GRF_GPIO6A_IOMUX[12]=1
i2s_sdo3	O	IO_I2Ssdo3_AU DIOgpio6a7	GRF_GPIO6A_IOMUX[14]=1

## 16.6 Application Notes

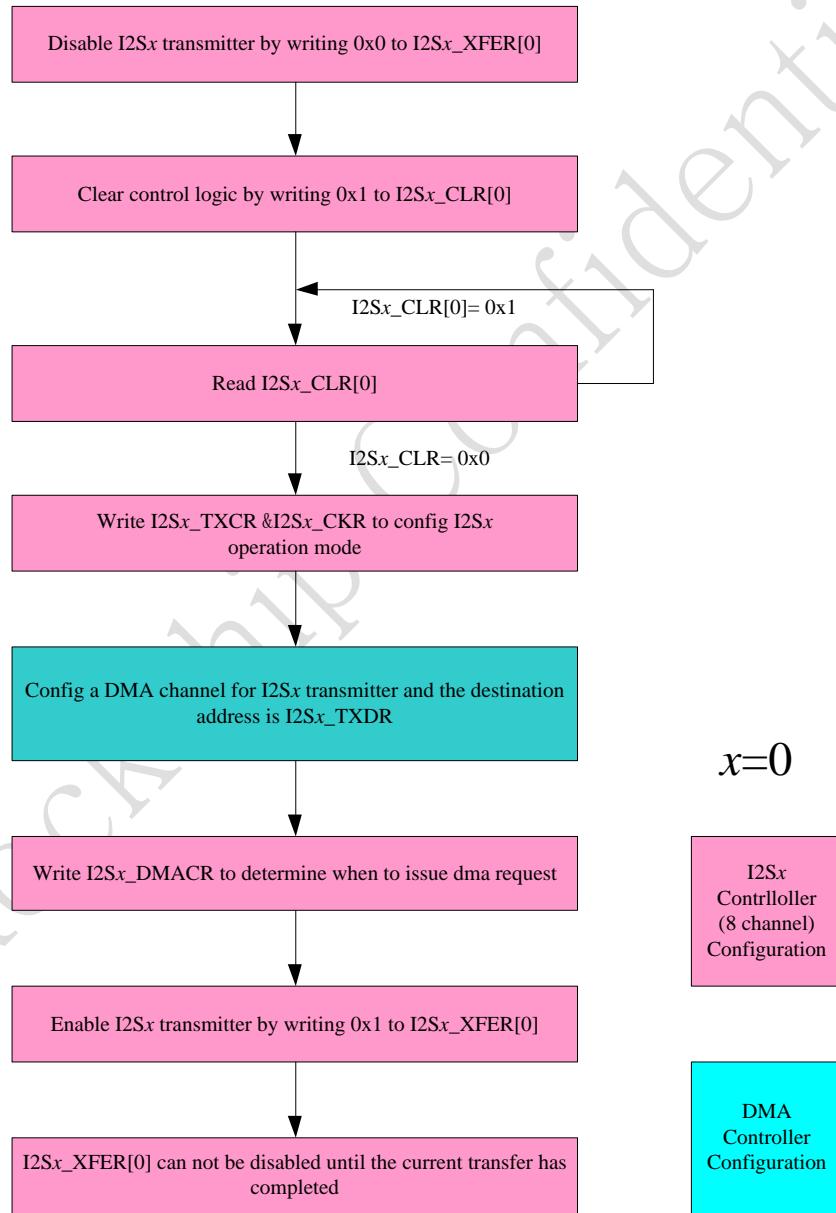


Fig. 16-11 I2S/PCM controller (8 channel) transmit operation flow chart

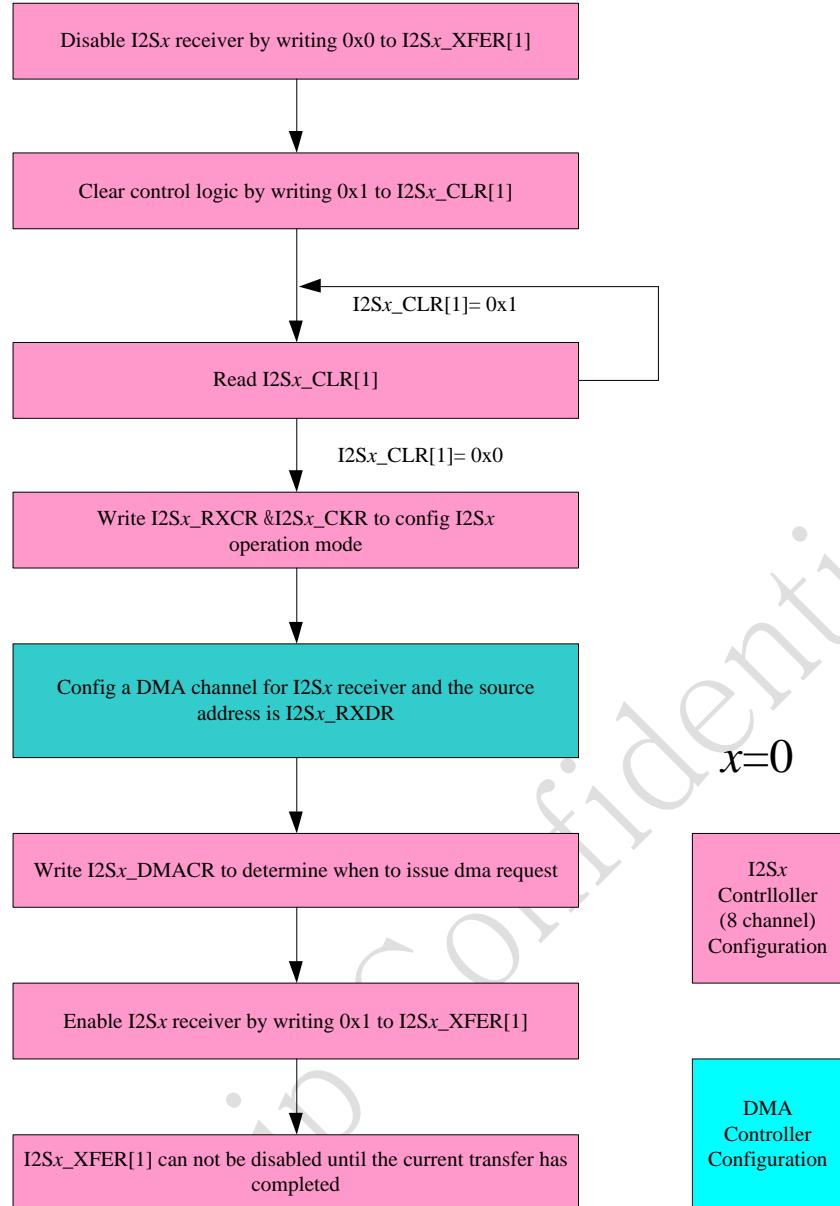


Fig. 16-12 I2S/PCM controller (8 channel) receive operation flow chart

## Chapter 17 SPDIF transmitter

### 17.1 Overview

The SPDIF transmitter is a self-clocking, serial, unidirectional interface for the interconnection of digital audio equipment for consumer and professional applications, using linear PCM coded audio samples.

It provides the basic structure of the interface. Separate documents define items specific to particular applications.

When used in a professional application, the interface is primarily intended to carry monophonic or stereophonic programmes, at a 48 kHz sampling frequency and with a resolution of up to 24bits per sample; it may alternatively be used to carry signals sampled at 32 kHz or 44.1 kHz.

When used in a consumer application, the interface is primarily intended to carry stereophonic programmes, with a resolution of up to 20 bits per sample, an extension to 24 bits per sample being possible.

When used for other purposes, the interface is primarily intended to carry audio data coded other than as linear PCM coded audio samples. Provision is also made to allow the interface to carry data related to computer software or signals coded using non-linear PCM. The format specification for these applications is not part of this standard.

In all cases, the clock references and auxiliary information are transmitted along with the programme.

- Supports one internal 32-bit wide and 32-location deep sample data buffer
- Supports two 16-bit audio data store together in one 32-bit wide location
- Supports AHB bus interface
- Supports biphase format stereo audio data output
- Supports DMA handshake interface and configurable DMA water level
- Supports sample data buffer empty, block terminate and user data interrupt
- Supports combine interrupt output
- Supports 16 to 31 bit audio data left or right justified in 32-bit wide sample data buffer
- Support 16, 20, 24 bits audio data transfer in linear PCM mode
- Support non-linear PCM transfer

### 17.2 Block Diagram

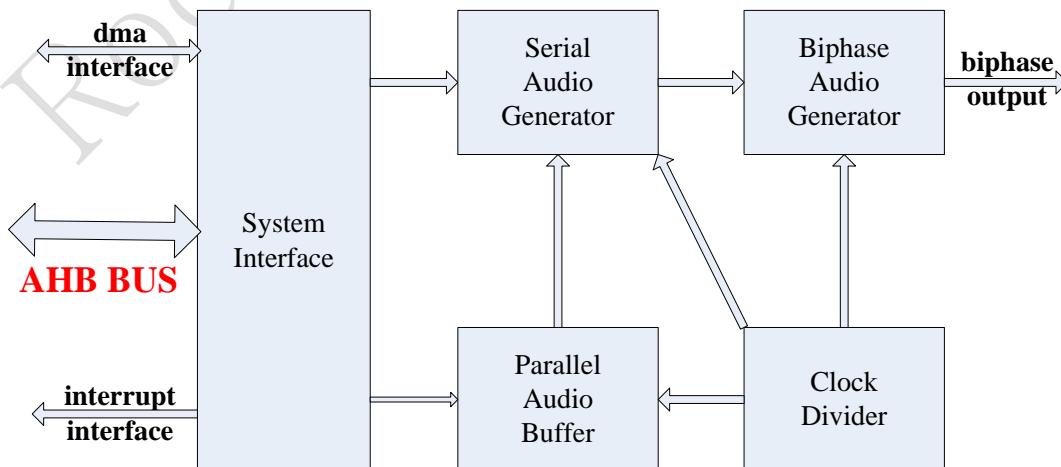


Fig. 17-1 SPDIF transmitter Block Diagram

#### System Interface

The system interface implements the AHB slave operation. It contains not only control registers of transmitters and receiver inside but also interrupt and DMA handshake interface.

### Clock Divider

The Clock Divider implements clock generation function. The input source clock to the module is MCLK, and by the divider of the module, the clock divider generates work clock for digital audio data transformation.

### Parallel Audio Buffer

The Parallel Audio Buffer is the buffer to store transmitted audio data. The size of the FIFO is 32bits x 32.

### Serial Audio Converter

The Serial Audio Converter reads parallel audio data from the Parallel Audio Buffer and converts it to serial audio data.

### Biphase Audio Generator

The Biphase Audio Generator reads serial audio data from the Serial Audio Converter and generates biphase audio data based on IEC-60958 standard.

## 17.3 Function description

### 17.3.1 Frame Format

A frame is uniquely composed of two sub-frames. For linear coded audio applications, the rate of transmission of frames corresponds exactly to the source sampling frequency.

In the 2-channel operation mode, the samples taken from both channels are transmitted by time multiplexing in consecutive sub-frames. The first sub-frame(left channel in stereophonic operation and primary channel in monophonic operation) normally use preamble M. However, the preamble is changed to preamble B once every 192 frame to identify the start of the block structure used to organize the channel status information. The second sub-frame (right in stereophonic operation and secondary channel in monophonic operation) always use preamble W.

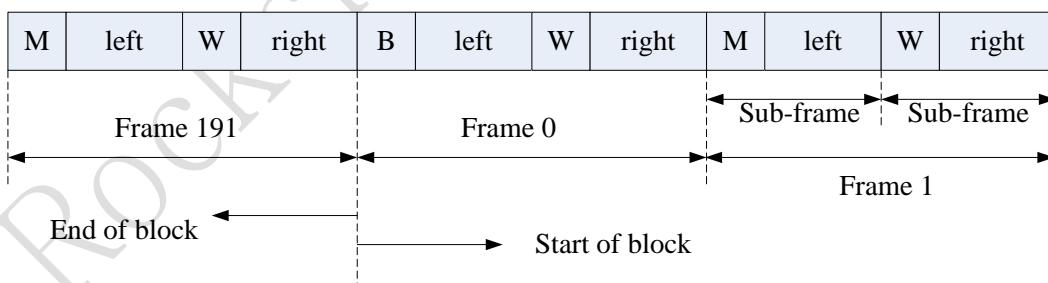


Fig. 17-2 SPDIF Frame Format

In the single channel operation mode in a professional application, the frame format is the same as in the 2-channel mode. Data is carried only in the first sub-frame and may be duplicated in the second sub-frame. If the second sub-frame is not carrying duplicate data, then time slot 28 (validity flag) shall be set to logical '1' (not valid).

### 17.3.2 Sub-frame Format

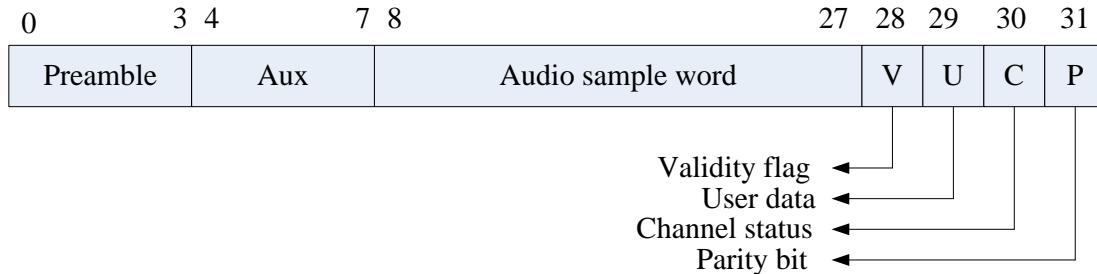


Fig. 17-3 SPDIF Sub-frame Format

Each sub-frame is divided into 32 time slots, numbered from 0 to 31. Time slot 0 to 3 carries one of the three permitted preambles. Time slot 4 to 27 carry the audio sample word in linear 2's complement representation. The MSB is carried by time slot 27. When a 24-bit coding range is used, the LSB is in time slot 4. When a 20-bit coding range is used, time slot 8 to 27 carry the audio sample word with the LSB in time slot 8. Time slot 4 to 7 may be used for other application. Under these circumstances, the bits in the time slot 4 to 7 are designated auxiliary sample bits.

If the source provides fewer bits than the interface allows (either 24 or 20), the unused LSBs are set to a logical '0'. For a non-linear PCM audio application or a data application the main data field may carry any other information. Time slot 28 carries the validity flag associated with the main data field. Time slot 29 carries 1 bit of the user data associated with the audio channel transmitted in the same sub-frame. Time slot 30 carries one bit of the channel status words associated with the main data field channel transmitted in the same sub-frame. Time slot 31 carries a parity bit such that time slots 4 to 31 inclusive carries an even number of ones and an even number of zeros.

### 17.3.3 Channel Coding

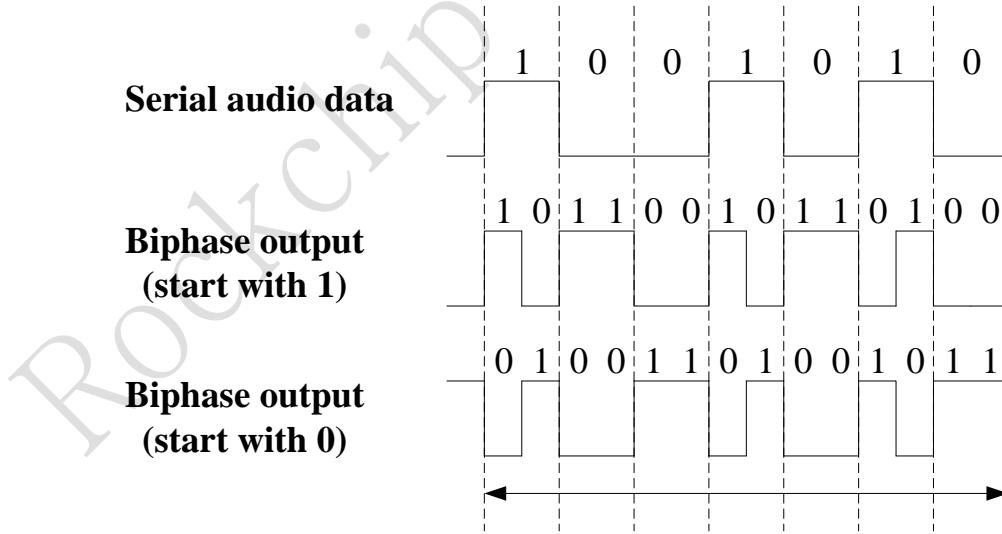


Fig. 17-4 SPDIF Channel Coding

To minimize the direct current component on the transmission line, to facilitate clock recovery from the data stream and to make the interface insensitive to the polarity of connections, time slots 4 to 31 are encoded in biphase-mark.

Each bit to be transmitted is represented by a symbol comprising two consecutive binary states. The first state of a symbol is always different from the second state of the previous symbol. The second state of the symbol is identical to the first if the bit to be transmitted is logical '0'. However, it is different from the first if the bit is logical '1'.

### 17.3.4 Preamble

Preambles are specific patterns providing synchronization and identification of the sub-frames and blocks.

To achieve synchronization within one sampling period and to make this process completely reliable, these patterns violate the biphase-mark code rules, thereby avoiding the possibility of data imitating the preambles.

A set of three preambles is used. These preambles are transmitted in the time allocated to four time slots (time slots 0 to 3) and are represented by eight successive states. The first state of the preamble is always different from the second state of the previous symbol.

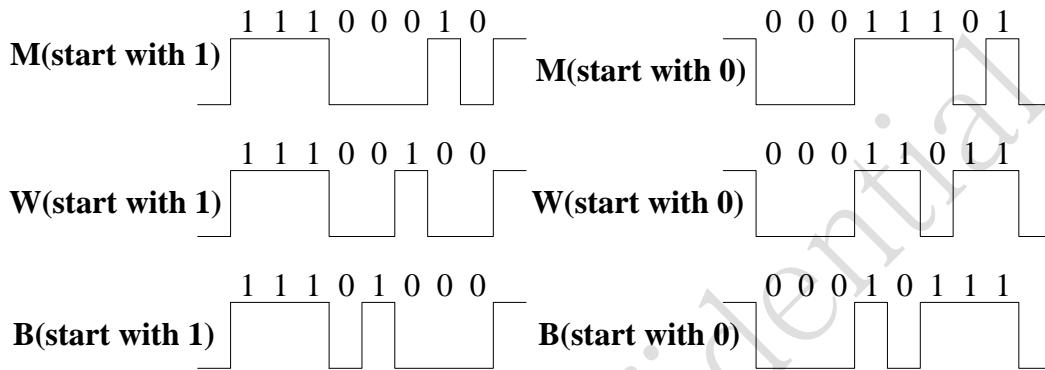


Fig. 17-5 SPDIF Preamble

Like biphase code, these preambles are dc free and provide clock recovery. They differ in at least two states from any valid biphase sequence.

### 17.3.5 NON-LINEAR PCM ENCODED SOURCE(IEC 61937)

The non-linear PCM encoded audio bitstream is transferred using the basic 16-bit data area of the IEC 60958 subframes, i.e. in time slots 12 to 27. Each IEC 60958 frame transfers 32-bit of the non-PCM data in consumer application mode.

If the SPDIF bitstream conveys linear PCM audio, the symbol frequency is 64 times the PCM sampling frequency (32 time slots per PCM sample times two channels). If a non-linear PCM encoded audio bitstream is conveyed by the interface, the symbol frequency is 64 times the sampling rate of the encoded audio within that bitstream. But in the case where a non-linear PCM encoded audio bitstream is conveyed by the interface containing audio with low sampling frequency, the symbol frequency is 128 times the sampling rate of the encoded audio within that bitstream.

Each data burst contains a burst-preamble consisting of four 16-bit words (Pa, Pb, Pc, Pd), followed by the burstpayload which contains data of an encoded audio frame.

The burst-preamble consists of four mandatory fields. Pa and Pb represent a synchronization word; Pc gives information about the type of data and some information/control for the receiver; Pd gives the length of the burstpayload, the number of bits or number of bytes according to data-type.

The four preamble words are contained in two sequential SPDIF frames. The frame beginning the data-burst contains preamble word Pa in subframe 0 and Pb in subframe 1. The next frame contains Pc in subframe 0 and Pd in subframe 1. When placed into a SPDIF subframe, the MSB of a 16-bit burst-preamble is placed into time slot 27 and the LSB is placed into time slot 12.

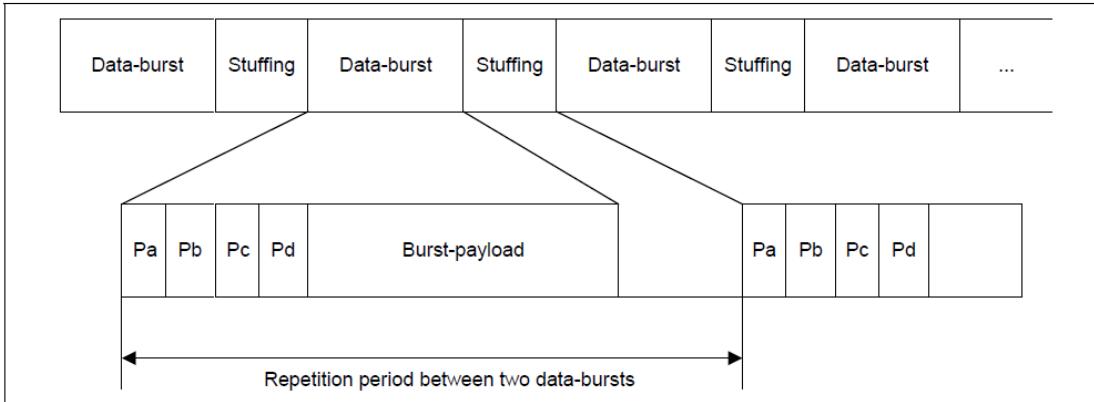


Fig. 17-6 Format of Data-burst

## 17.4 Register description

### 17.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
SPDIF_CFGR	0x0000	W	0x00000000	Transfer Configuration Register
SPDIF_SDBLR	0x0004	W	0x00000000	Sample Date Buffer Level Register
SPDIF_DMCR	0x0008	W	0x00000000	DMA Control Register
SPDIF_INTCR	0x000c	W	0x00000000	Interrupt Control Register
SPDIF_INTSR	0x0010	W	0x00000000	Interrupt Status Register
SPDIF_XFER	0x0018	W	0x00000000	Transfer Start Register
SPDIF_SMPDR	0x0020	W	0x00000000	Sample Data Register
SPDIF_VLDFRN	0x0060	W	0x00000000	Validity Flag Register n
SPDIF_USRDRN	0x0090	W	0x00000000	User Data Register n
SPDIF_CHNSRN	0x00c0	W	0x00000000	Channel Status Register n
SPDIF_BURTSINFO	0x0100	W	0x00000000	Channel Burst Info Register
SPDIF_REPETITION	0x0104	W	0x00000000	Channel Repetition Register
SPDIF_BURTSINFO_SHD	0x0108	W	0x00000000	Shadow Channel Burst Info Register
SPDIF_REPETITION_SHD	0x010c	W	0x00000000	Shadow Channel Repetition Register
SPDIF_USRDR_SHDn	0x0190	W	0x00000000	Shadow User Data Register n

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

### 17.4.2 Detail Register Description

#### SPDIF\_CFGR

Address: Operational Base + offset (0x0000)

Transfer Configuration Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23:16	RW	0x00	MCD mclk divider Fmclk/Fsdo This parameter can be caculated by Fmclk/(Fs*128). Fs=the sample frequency be wanted
15:10	RO	0x0	reserved
9	RW	0x0	PRE_CHANGE Preamble Change The bit only is valid when set to non-linear PCM mode; 0: the Preamble will change when block finish; 1: the Preamble will change every 192 frames just like the linear PCM mode.
8	RW	0x0	PCMTYPE PCM type 0: linear PCM 1: non-linear PCM
7	WO	0x0	CLR mclk domain logic clear Write 1 to clear mclk domain logic. Read return zero.
6	RW	0x0	CSE Channel status enable 0: disable 1: enable The bit should be set to 1 when the channel conveys non-linear PCM
5	RW	0x0	UDE User data enable 0: disable 1: enable
4	RW	0x0	VFE Validity flag enable 0: disable 1: enable
3	RW	0x0	ADJ audio data justified 0: Right justified 1: Left justified

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x0	HWT Halfword word transform enable 0: disable 1: enable It is valid only when the valid data width is 16bit.
1:0	RW	0x0	VDW Valid data width 00: 16bit 01: 20bit 10: 24bit 11: reserved The valid data width is 16bit only for non-linear PCM

**SPDIF\_SDBLR**

Address: Operational Base + offset (0x0004)

Sample Date Buffer Level Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RW	0x00	SDBLR Sample Date Buffer Level Register Contains the number of valid data entries in the sample data buffer.

**SPDIF\_DMACR**

Address: Operational Base + offset (0x0008)

DMA Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x0	TDE Transmit DMA Enable 0: Transmit DMA disabled 1: Transmit DMA enabled
4:0	RW	0x00	TDL Transmit Data Level This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the Sample Date Buffer is equal to or below this field value

**SPDIF\_INTCR**

Address: Operational Base + offset (0x000c)

## Interrupt Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:18	RO	0x0	reserved
17	W1C	0x0	UDTIC User Data Interrupt Clear Write '1' to clear the user data interrupt.
16	W1C	0x0	BTTIC Block/Data burst transfer finish interrupt clear Write 1 to clear the interrupt.
15:10	RO	0x0	reserved
9:5	RW	0x00	SDBT Sample Date Buffer Threshold Sample Date Buffer Threshold for empty interrupt
4	RW	0x0	SDBEIE Sample Date Buffer empty interrupt enable 0: disable 1: enable
3	RW	0x0	BTTIE Block transfer/repetition period end interrupt enable When enabled, an interrupt will be asserted when the block transfer is finished if the channel conveys linear PCM or when the repetition period is reached if the channel conveys non-linear PCM. 0: disable 1: enable
2	RW	0x0	UDTIE User Data Interrupt 0: disable 1: enable If enabled, an interrupt will be asserted when the content of the user data register is fed into the corresponding shadow register
1:0	RO	0x0	reserved

**SPDIF\_INTSR**

Address: Operational Base + offset (0x0010)

## Interrupt Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4	RW	0x0	SDBEIS Sample Date Buffer empty interrupt status 0: inactive 1: active

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	BTTIS Block/Data burst transfer interrupt status 0: inactive 1: active
2	RW	0x0	UDTIS User Data Interrupt Status 0: inactive 1: active
1:0	RO	0x0	reserved

**SPDIF\_XFER**

Address: Operational Base + offset (0x0018)

Transfer Start Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	XFER Transfer Start Register Transfer Start Register

**SPDIF\_SMPDR**

Address: Operational Base + offset (0x0020)

Sample Data Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	SMPDR Sample Data Register Sample Data Register

**SPDIF\_VLDFRn**

Address: Operational Base + offset (0x0060)

Validity Flag Register n

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	VLDFR_SUB_1 Validity Flag Subframe 1 Validity Flag Register 0
15:0	RW	0x0000	VLDFR_SUB_0 Validity Flag Subframe 0 Validity Flag for Subframe 0

**SPDIF\_USRDRn**

Address: Operational Base + offset (0x0090)

User Data Register n

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	USR_SUB_1 User Data Subframe 1 User Data Bit for Subframe 1
15:0	RW	0x0000	USR_SUB_0 User Data Subframe 0 User Data Bit for Subframe 0

**SPDIF\_CHNSRn**

Address: Operational Base + offset (0x00c0)

Channel Status Register n

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	CHNSR_SUB_1 Channel Status Subframe 1 Channel Status Bit for Subframe 1
15:0	RW	0x0000	CHNSR_SUB_0 Channel Status Subframe 0 Channel Status Bit for Subframe 0

**SPDIF\_BURTSINFO**

Address: Operational Base + offset (0x0100)

Channel Burst Info Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	PD pd Preamble Pd for non-linear pcm, indicating the length of burst payload in unit of bytes or bits.
15:13	RW	0x0	BSNUM Bitstream Number This field indicates the bitstream number. Usually the birstream number is 0.
12:8	RW	0x00	DATAINFO Data-type-dependent info This field gives the data-type-dependent info
7	RW	0x0	ERRFLAG Error Flag 0: indicates a valid burst-payload 1: indicates that the burst-payload may contain errors

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6:0	RW	0x00	<p>DATATYPE Data type 0000000: null data 0000001: AC-3 data 0000011: Pause data 0000100: MPEG-1 layer 1 data 0000101: MPEG-1 layer 2 or 3 data or MPEG-2 without extension 0000110: MPEG-2 data with extension 0000111: MPEG-2 AAC 0001000: MPEG-2, layer-1 low sampling frequency 0001001: MPEG-2, layer-2 low sampling frequency 0001010: MPEG-2, layer-3 low sampling frequency 0001011: DTS type I 0001100: DTS type II 0001101: DTS type III 0001110: ATRAC 0001111: ATRAC 2/3 0010000: ATRAC-X 0010001: DTS type IV 0010010: WMA professional type I 0110010: WMA professional type II 1010010: WMA professional type III 1110010: WMA professional type IV 0010011: MPEG-2 AAC low sampling frequency 0110011: MPEG-2 AAC low sampling frequency 1010011: MPEG-2 AAC low sampling frequency 1110011: MPEG-2 AAC low sampling frequency 0010100: MPEG-4 AAC 0110100: MPEG-4 AAC 1010100: MPEG-4 AAC 1110100: MPEG-4 AAC 0010101: Enhanced AC-3 0010110: MAT others: reserved </p>

**SPDIF\_REPEATION**

Address: Operational Base + offset (0x0104)  
Channel Repetition Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	REPETITION Repetition This define the repetition period when the channel conveys non-linear PCM

**SPDIF\_BURTSINFO\_SHD**

Address: Operational Base + offset (0x0108)

Shadow Channel Burst Info Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	PD pd Preamble Pd for non-linear pcm, indicating the length of burst payload in unit of bytes or bits.
15:13	RO	0x0	BSNUM Bitstream Number This field indicates the bitstream number. Usually the birstream number is 0.
12:8	RO	0x00	DATAINFO Data-type-dependent info This field gives the data-type-dependent info
7	RO	0x0	ERRFLAG Error Flag 0: indicates a valid burst-payload 1: indicates that the burst-payload may contain errors

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6:0	RO	0x00	<p>DATATYPE Data type 0000000: null data 0000001: AC-3 data 0000011: Pause data 0000100: MPEG-1 layer 1 data 0000101: MPEG-1 layer 2 or 3 data or MPEG-2 without extension 0000110: MPEG-2 data with extension 0000111: MPEG-2 AAC 0001000: MPEG-2, layer-1 low sampling frequency 0001001: MPEG-2, layer-2 low sampling frequency 0001010: MPEG-2, layer-3 low sampling frequency 0001011: DTS type I 0001100: DTS type II 0001101: DTS type III 0001110: ATRAC 0001111: ATRAC 2/3 0010000: ATRAC-X 0010001: DTS type IV 0010010: WMA professional type I 0110010: WMA professional type II 1010010: WMA professional type III 1110010: WMA professional type IV 0010011: MPEG-2 AAC low sampling frequency 0110011: MPEG-2 AAC low sampling frequency 1010011: MPEG-2 AAC low sampling frequency 1110011: MPEG-2 AAC low sampling frequency 0010100: MPEG-4 AAC 0110100: MPEG-4 AAC 1010100: MPEG-4 AAC 1110100: MPEG-4 AAC 0010101: Enhanced AC-3 0010110: MAT others: reserved </p>

**SPDIF\_REPEATION\_SHD**

Address: Operational Base + offset (0x010c)

Shadow Channel Repetition Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RO	0x0000	REPETITION Repetition This register provides the repetition of the bitstream when channel conveys non-linear PCM. In the design, it is define the length between Pa of the two consecutive data-burst. For the same audio format, the definition is different. Please convert the actual repetition in order to comply with the design.

**SPDIF\_USRDR\_SHDn**

Address: Operational Base + offset (0x0190)

Shadow User Data Register n

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	USR_SUB_1 User Data Subframe 1 User Data Bit for Subframe 1
15:0	RO	0x0000	USR_SUB_0 User Data Subframe 0 User Data Bit for Subframe 0

**17.5 Interface description**

Table 17-1 IOMUX Setting

<b>Module Pin</b>	<b>IO</b>	<b>Pad Name</b>	<b>IOMUX Setting</b>
spdif_tx	O	SPDIFtx_AUDIOgpio6b3	GPIO6B_IOMUX[7:6]= 2'b01

Notes: 1. O=output

## 17.6 Application Notes

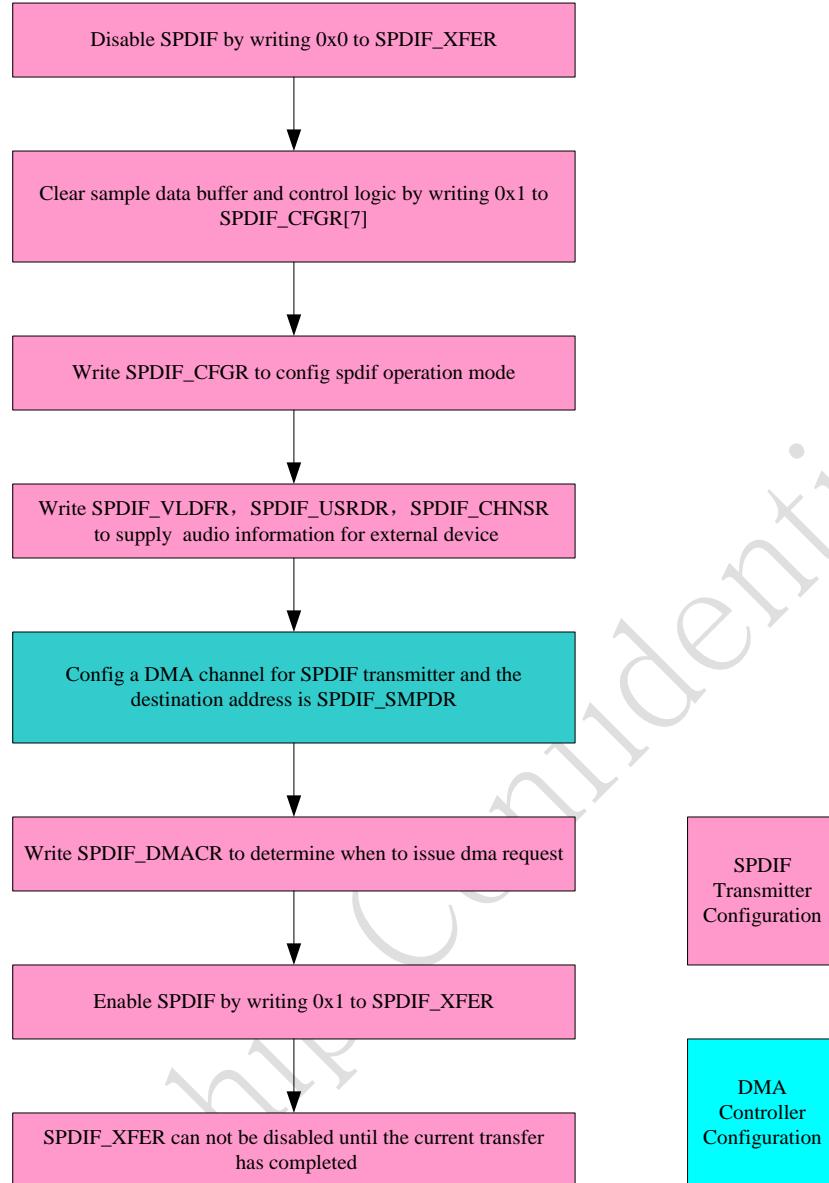


Fig. 17-7 SPDIF transmitter operation flow chart

### 17.6.1 Channel Status Bit and Validity Flag Bit

Normally the channel status bits and validity flag bits are not necessarily updated frequently. If it is desired to change the channel status bits or validity flag, please write to the corresponding register after a block terminate interrupt is asserted. The new value will take effect immediately.

### 17.6.2 User Data Bit

As the user data bits are updated frequently, the design takes use of the shadow register mechanism to store and convey the user data bit. When the SPDIF interface is disabled, the values of the shadow user data registers keeps the same with the corresponding user data registers. After the SPDIF starts, any change of the user data register will not go to the corresponding shadow user data registers until an user data interrupt is asserted.

Therefore before the SPDIF transfer starts, prepare the first 384 user data bits by writing them to the SPDIF\_USRDR registers. After the SPDIF transfer starts, writing the second 384

user data bits to the SPDIF\_USRDR registers. Then wait for the assertion of user data interrupt. The second 384 user data bits goes to the shadow registers, and then third 384 user bits are written to SPDIF\_USRDR.

### **17.6.3 Burst Info and Repetition**

The shadow register mechanism is also applied to the data of burst info and repetition as the user data. The difference is that the update of shadow register will be taken after assertion of the block terminate interrupt.

It is important to note that the repetition defined in the design is a little different from the repetition defined in IEC-61957. The repetition is always defined as the length (measured in IEC-60958 frame) between Pa of two consecutive data-bursts. Therefore the user needs to calculation the new repetition value if the definition of the repetition is different for some audio formats such as AC-3.

## Chapter 18 USB OTG 2.0

### 18.1 Overview

USB OTG 2.0 is a Dual-Role Device controller, which supports both device and host functions and is fully compliant with OTG Supplement to USB2.0 specification, and support high-speed (480Mbps), full-speed (12Mbps), low-speed (1.5Mbps) transfer.

USB OTG 2.0 is optimized for portable electronic devices, point-to-point applications (no hub, direct connection to device) and multi-point applications to devices. USB OTG 2.0 interface supports both device and host functions and is fully compliant with OTG Supplement to USB2.0 specification, and support high-speed (480Mbps), full-speed (12Mbps), low-speed (1.5Mbps) transfer. It is optimized for portable electronic device, point-to-point applications (no hub, direct connection to device) and multi-point applications to devices.

The USB OTG 2.0 supports following features:

- Compliant with the OTG Supplement to the USB2.0 Specification
- Operates in High-Speed and Full-Speed mode
- Support 9 channels in host mode
- 9 Device mode endpoints in addition to control endpoint 0, 4 in, 3 out and 2 IN/OUT
- Built-in one 1024x35 bits FIFO
- Internal DMA with scatter/gather function
- Supports packet-based, dynamic FIFO memory allocation for endpoints for flexible, efficient use of RAM
- Support dynamic FIFO sizing
- Support Battery Charge
- Support UART Bypass function

### 18.2 Block Diagram

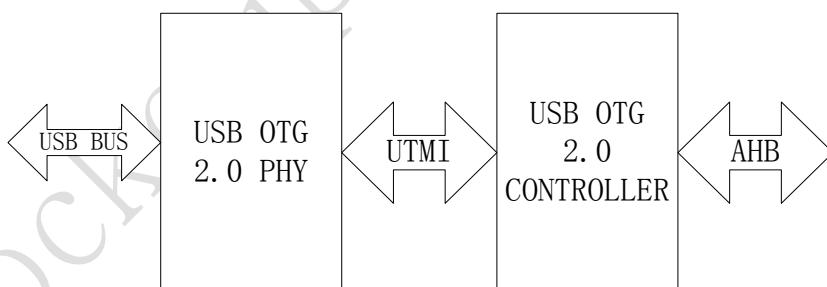


Fig. 18-1 USB OTG 2.0 Architecture

Fig.19-1 shows the architecture of USB OTG 2.0. It is broken up into two separate units: USB OTG 2.0 controller and USB OTG 2.0 PHY. The two units are interconnected with UTMI interface.

#### 18.2.1 USB OTG 2.0 Controller Function

The USB OTG 2.0 Controller controls SIE (Serial Interface Engine) logic, the endpoint logic, the channel logic and the internal DMA logic.

The SIE logic contains the USB PID and address recognition logic, and other sequencing and state machine logic to handle USB packets and transactions. Generally the SIE Logic is required for any USB implementation while the number and types of endpoints will vary as function of application and performance requirements.

The endpoint logic contains the endpoint specific logic: endpoint number recognition, FIFOs

and FIFO control, etc.

The channel Logic contains the channel tasks schedule, FIFOs and FIFO control, etc.

The internal DMA logic controls data transaction between system memory and USB FIFOs.

### 18.2.2 USB OTG 2.0 PHY Function

The USB OTG 2.0 PHY handles the low level USB protocol and signaling. This includes features such as; data serialization and deserialization, bit stuffing and clock recovery and synchronization. The primary focus of this block is to shift the clock domain of the data from the USB 2.0 rate to the frequency of UTMI clock which is 30MHz.

### 18.2.3 UTMI Interface

- Transmit

Transmit must be asserted to enable any transmissions.

The USB OTG2.0 CONTROLLER asserts TXValid to begin a transmission and negates TXValid to end a transmission. After the USB OTG2.0 CONTROLLER asserts TXValid it can assume that the transmission has started when it detects TXReady asserted.

The USB OTG2.0 CONTROLLER assumes that the USB OTG2.0 PHY has consumed a data byte if TXReady and TXValid are asserted.

The USB OTG2.0 CONTROLLER must have valid packet information (PID) asserted on the Data In bus coincident with the assertion of TXValid. Depending on the USB OTG2.0 PHY implementation, TXReady may be asserted by the Transmit State Machine as soon as one CLK after the assertion of TXValid. TXValid and TXReady are sampled on the rising edge of CLK.

The Transmit State Machine does NOT automatically generate Packet ID's (PIDs) or CRC. When transmitting, the USB OTG2.0 CONTROLLER is always expected to present a PID as the first byte of the data stream and if appropriate, CRC as the last bytes of the data stream.

The USB OTG2.0 CONTROLLER must use LineState to verify a Bus Idle condition before asserting TXValid in the TX Wait state.

The state of TXReady in the TX Wait and Send SYNC states is undefined. An MTU implementation may prepare for the next transmission immediately after the Send EOP state and assert TXReady in the TX Wait state. An MTU implementation may also assert TXReady in the Send SYNC state. The first assertion of TXReady is Macrocell implementation dependent. The USB OTG2.0 CONTROLLER must prepare DataIn for the first byte to be transmitted before asserting TXValid.

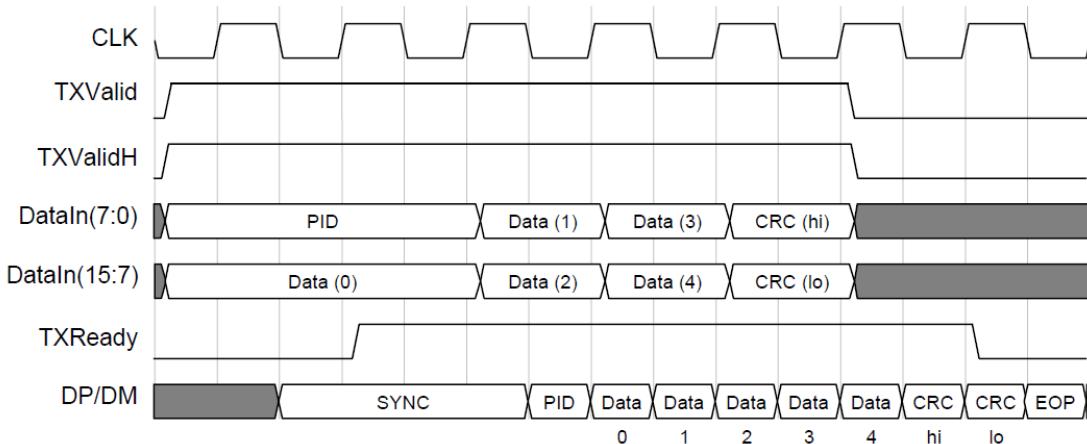


Fig. 18-2 UTMI interface – Transmit timing for a data packet

- Receive

RXActive and RXValid are sampled on the rising edge of CLK.

In the RX Wait state the receiver is always looking for SYNC.

The USB OTG 2.0 PHY asserts RXActive when SYNC is detected (Strip SYNC state).

The USB OTG 2.0 PHY negates RXActive when an EOP is detected (Strip EOP state).

When RxActive is asserted, RXValid will be asserted if the RX Holding Register is full.

RXValid will be negated if the RX Holding Register was not loaded during the previous byte time.

This will occur if 8 stuffed bits have been accumulated.

The USB OTG2.0 Controller must be ready to consume a data byte if RXActive and RXValid are asserted (RX Data state).

In FS mode, if a bit stuff error is detected then the Receive State Machine will negate RXActive and RXValid, and return to the RX Wait state.

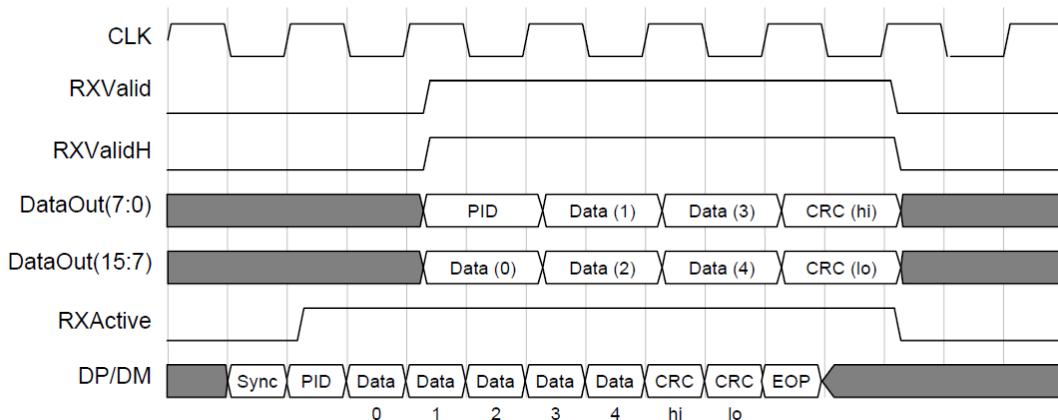


Fig. 18-3 UTMI interface – Receive timing for a data packet

## 18.3 USB OTG2.0 Controller

Fig.19-4 shows the main components and flow of the USB OTG 2.0 controller system.

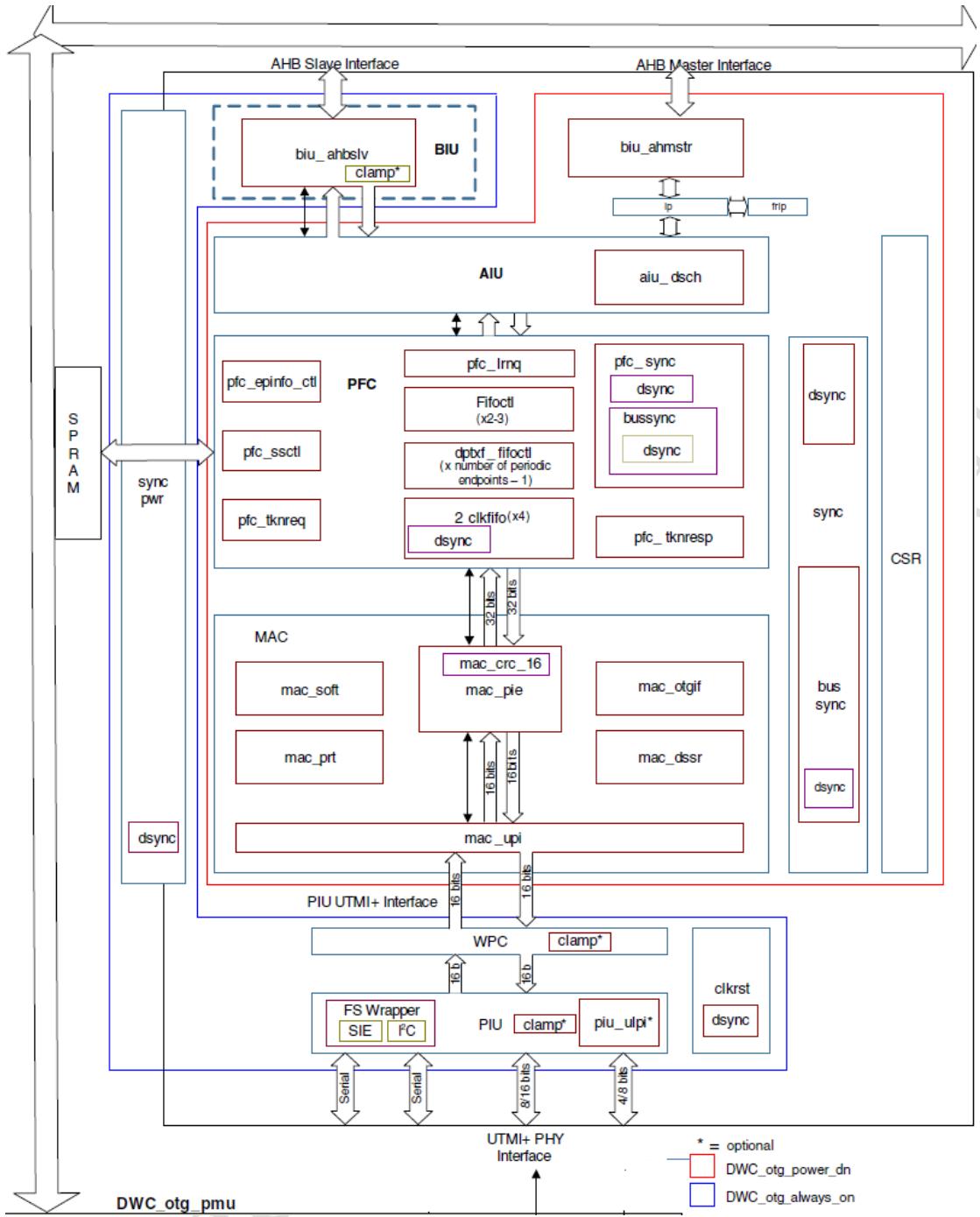


Fig. 18-4 USB OTG2.0 Controller Architecture

### 1). AHB Slave Bus Interface Unit (BIUS)

The AHB Slave interface unit converts AHB cycles to CSR write/read, Data-FIFO read/write, and DFIFO push/pop signals.

### 2) Control and Status Registers (CSR)

The CSR block resides in the AHB clock domain, and contains all registers except the Power and Clock Gating Control Register (PCGCCTL) and bits 31:29 of the Core Interrupt register (GINTSTS).

### 3) Application Interface Unit (AIU)

The application Interface Unit (AIU) consists of the following interfaces:

- AHB Master

- AHB Slave
- Packet FIFO Controller
- Control and Status registers

#### 4). DMA Scheduler (DSCH)

This block is used only in DMA mode. It controls the transfer of data packets between the system memory and the USB OTG 2.0 Controller for both Internal and External DMA.

#### 5). Packet FIFO Controller (PFC)

Several FIFOs are used in Device and Host modes to store data inside the core before transmitting it on either the AHB or the USB. PFC connect the Data FIFO interface to an industry-standard, single-port synchronous SRAM. Address, write data, and control outputs are driven late by the USB OTG 2.0 Controller, but in time to meet the SRAM setup requirements. Input read data is expected late from the SRAM and registered inside the core before being used.

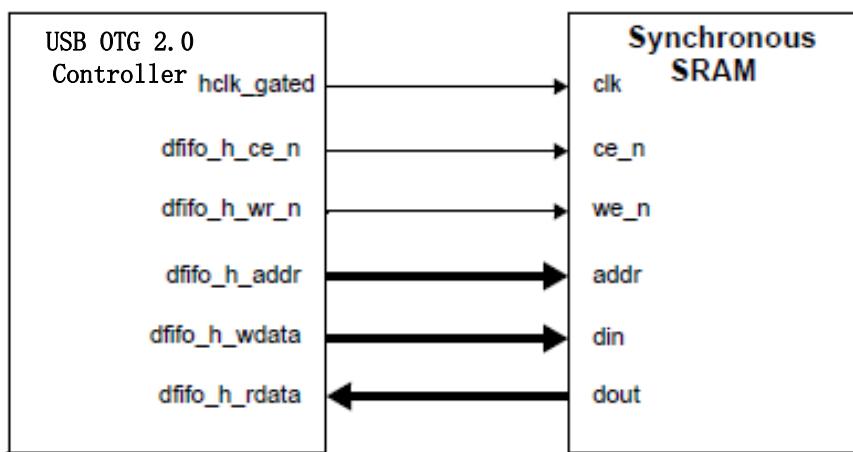


Fig. 18-5 DFIFO single-port synchronous SRAM interface

#### 6). Media Access Controller (MAC)

The Media Access Controller (MAC) module handles USB transactions, and device, host, and OTG protocols.

#### 7) PHY Interface Unit (PIU)

The core uses 16-bit UTMI+ Interface.

#### 8) Wakeup and Power Controller (WPC)

When the USB is suspended or the session is not valid, the PHY is driven into Suspend mode and the PHY clock is stopped to reduce PHY and the core power consumption. To reduce power consumption further, the core also supports AHB clock gating and partial power-down.

### 18.3.1 Host Architecture

The host uses one transmit FIFO for all non-periodic OUT transactions and one transmit FIFO for all periodic OUT transactions. These transmit FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over USB.

The host pipes the USB transactions through Request queues (one for periodic and one for non-periodic). Each entry in the Request - queue holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the requests are written into the queue determines the sequence of transactions on the USB. The host processes the periodic Request queue first, followed by the non-periodic Request queue, at the beginning of each (micro) frame.

The host uses one Receive-FIFO for all periodic and non-periodic transactions. The FIFO is used as a Receive-buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. The status of each packet received also goes into the FIFO. The status entry holds the IN channel number along with other information, such as received byte count and validity status, to perform a transaction on the AHB.

### 18.3.2 Device Architecture

The core uses Dedicated Transmit FIFO Operation. In this mode, there are individual transmit FIFOs for each IN endpoint.

The OTG device uses a single receive FIFO to receive the data for all the OUT endpoints. The receive FIFO holds the status of the received data packet, such as byte count, data PID and the validity of the received data. The DMA or the application reads the data out of the receive FIFO as it is received.

### 18.3.3 FIFO Mapping

- Fig.19-6 shows FIFO mapping in Host mode.

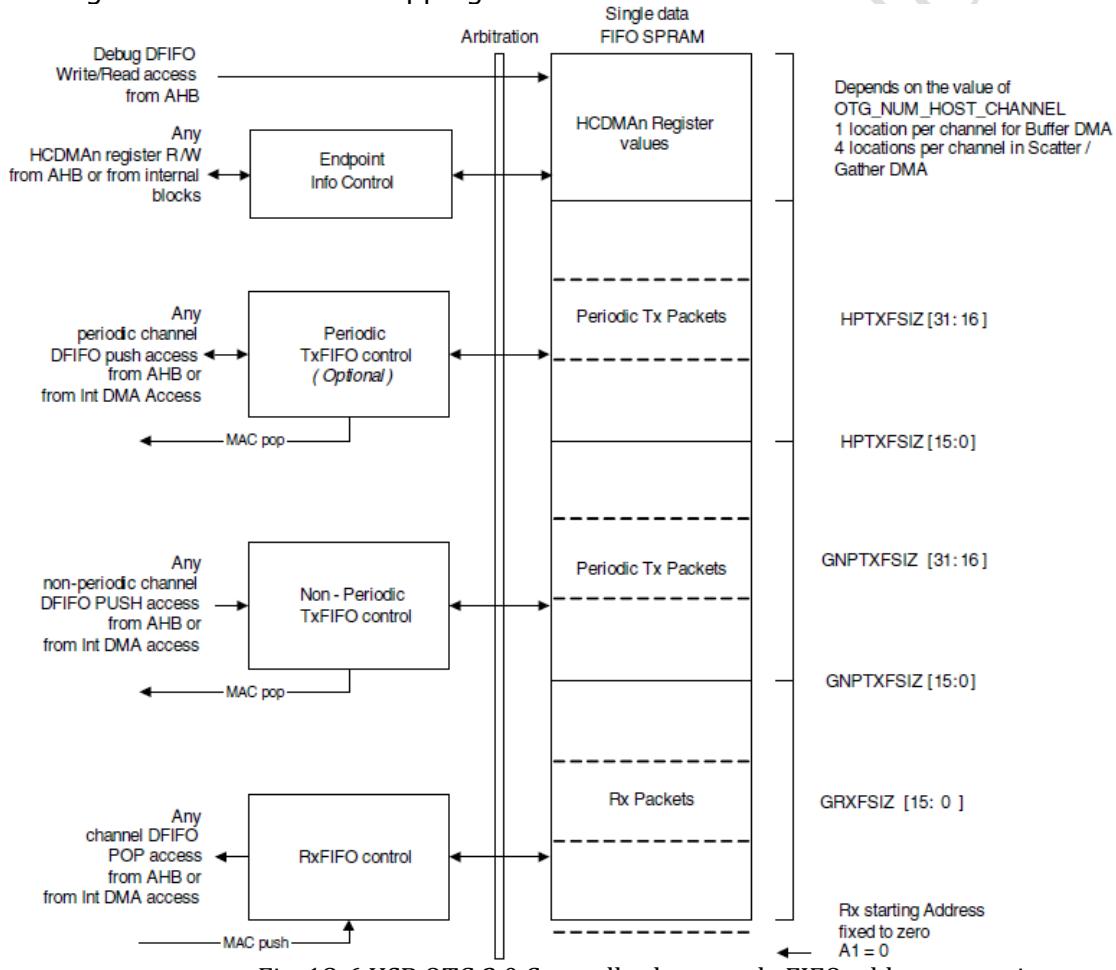


Fig. 18-6 USB OTG 2.0 Controller host mode FIFO address mapping

*Note: When the device is operating in Internal DMA mode, the last locations of the SPRAM are used to store the DMAADDR values for each channel.*

- Fig.19-7 shows FIFO mapping in Device mode.

When the device is operating in non-Descriptor Internal DMA mode, the last locations of the

SPRAM are used to store the DMAADDR values for each channel. When the device is operating in Descriptor mode, then the last locations of the SPRAM store the Base Descriptor address, Current Descriptor address, Current Buffer address, and status quadlet information for each endpoint direction.

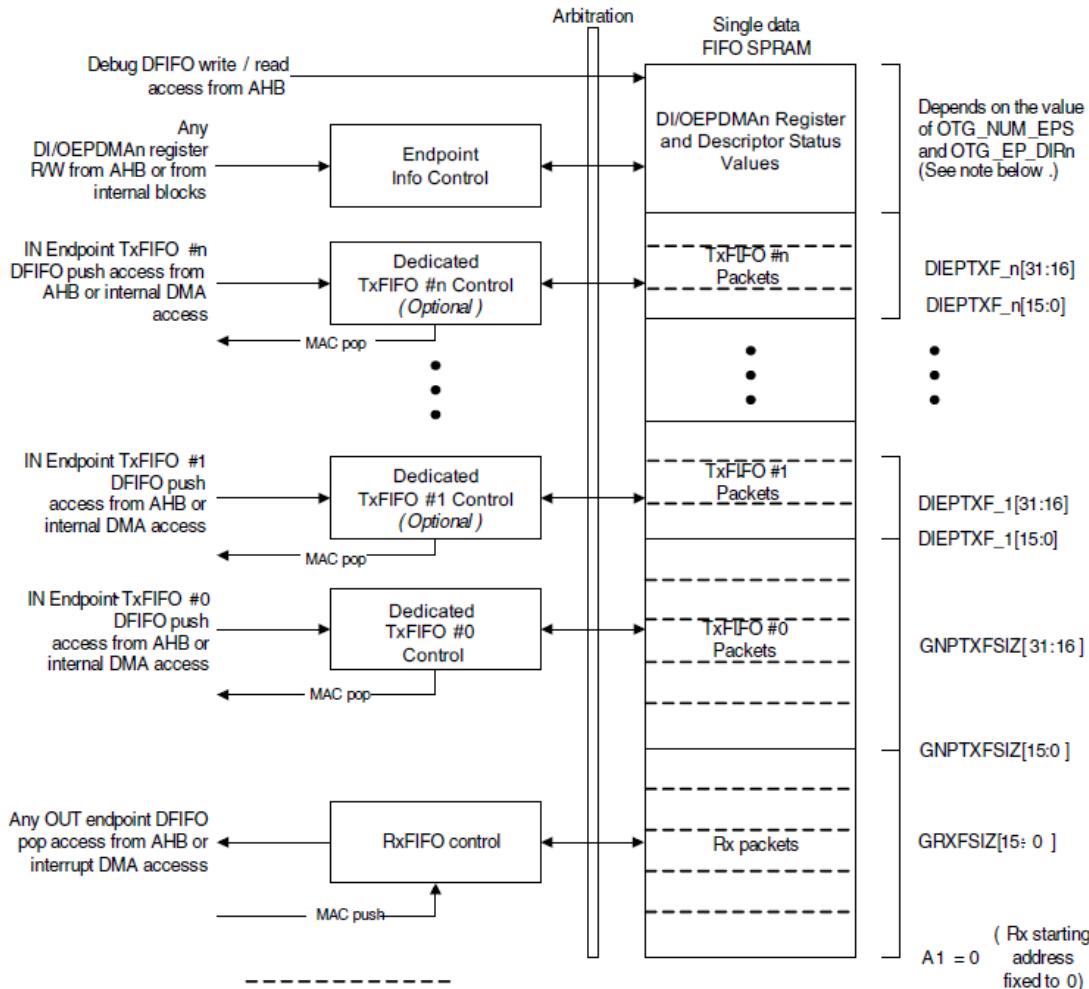


Fig. 18-7 USB OTG 2.0 Controller device mode FIFO address mapping

*Note: When the device is operating in non-Scatter Gather Internal DMA mode, the last locations of the SPRAM are used to store the DMAADDR values for each Endpoint (1 location per endpoint). When the device is operating in Scatter Gather mode, then the last locations of the SPRAM store the Base Descriptor address, Current Descriptor address, Current Buffer address, and status quadlet information for each endpoint direction (4 locations per Endpoint). If an Endpoint is bidirectional, then 4 locations will be used for IN, and another 4 for OUT).*

## 18.4 USB OTG2.0 PHY

The USB OTG 2.0 PHY connects a USB OTG controller to a USB system. It is a complete mixed-signal IP designed to implement OTG connectivity in a System-on-Chip (SOC) design targeted to a specific fabrication process using core and 2.5-V thick-oxide devices. The USB 2.0 PHY supports the USB2.0 480-Mbps protocol and data rate, and is backward compatible with the USB 1.1 1.5-Mbps and 12-Mbps protocol and data rates.

### 18.4.1 Block Diagram

Fig.19-8 shows the USB OTG 2.0 PHY functional block diagram for a one-port macro.

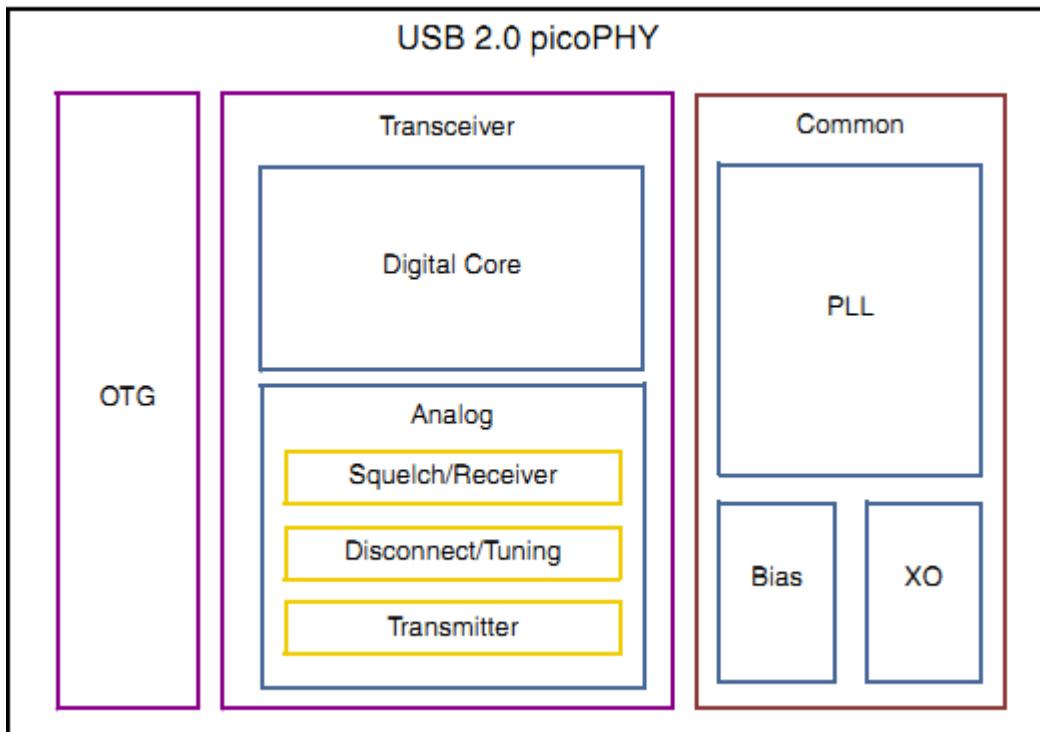


Fig. 18-8 USB OTG 2.0 PHY Architecture

The USB OTG 2.0 PHY consists of three basic components: the Common block, Transceiver block, and OTG block.

- Common block: This block contains design components that can be reused for multiple transceivers.
- Transceiver block: This block contains the bulk of USB OTG 2.0 PHY circuitry for data processing and transfers.
- OTG block: This block enables A-devices and B-devices to initiate the Session Request Protocol (SRP), and dual-Role devices to initiate the Host Negotiation Protocol (HNP).

#### 18.4.2 Powering Up and Powering Down

- Powering UP

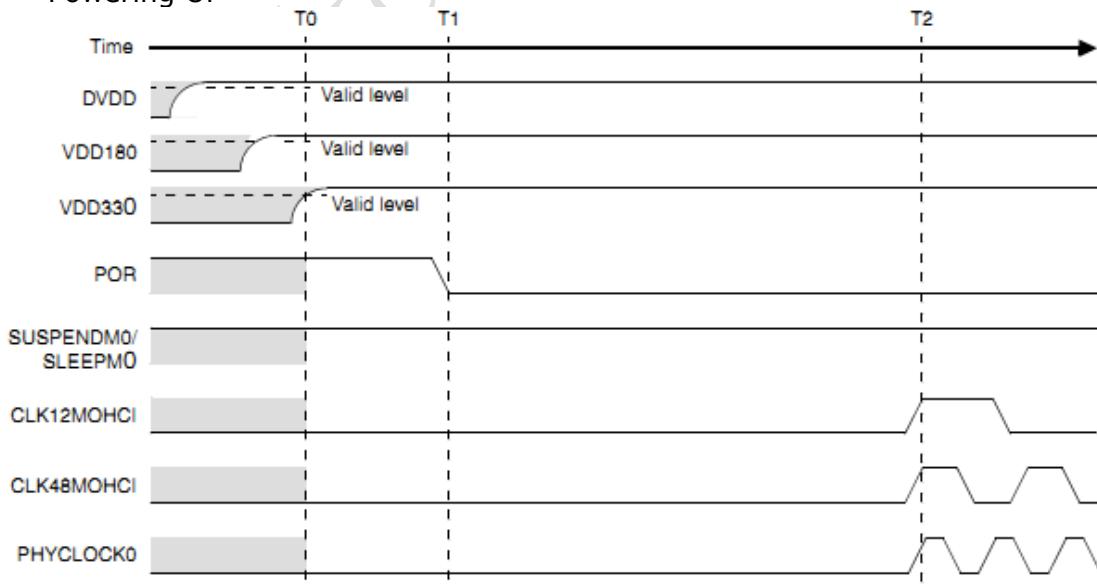


Fig. 18-9 USB OTG 2.0 PHY power supply and power up sequence

Table 18-1 USB OTG 2.0 PHY power supply timing parameter

Timing Parameter	Description	Value
T0	Power-on reset (POR) is initiated.	0 (reference)
T1	T1 indicates when POR can be set to 1'b0. (To provide examples, values for T2 and T3 are also shown where $T1 = T0 + 30 \mu s$ .) In general, T1 must be $\geq T0 + 10 \mu s$ .	$T0 + 10 \mu s \leq T1$
T2	T2 indicates when PHYCLOCK0, CLK48MOHCl, and CLK12MOHCl are available at the macro output, based on the USB 2.0 picoPHY reference clock source.	<p>Crystal:</p> <ul style="list-style-type: none"> <li>▪ When <math>T1 = T0 + 10 \mu s</math>: <math>T2 &lt; T1 + 805 \mu s = T0 + 815 \mu s</math></li> <li>▪ When <math>T1 = T0 + 30 \mu s</math>: <math>T2 &lt; T1 + 805 \mu s = T0 + 835 \mu s</math></li> </ul> <p>External board clock or CLKCORE:</p> <ul style="list-style-type: none"> <li>▪ When <math>T1 = T0 + 10 \mu s</math>: <math>T2 &lt; T1 + 45 \mu s = T0 + 55 \mu s</math></li> <li>▪ When <math>T1 = T0 + 30 \mu s</math>: <math>T2 &lt; T1 + 45 \mu s = T0 + 75 \mu s</math></li> </ul>

### 18.4.3 Removing Power Supplies for Power Saving

There is no requirement on the power-down sequence for the USB groups.

Customers can decide which voltage to be down first based on the application, it is recommended to keep the time between collapsing of power supplies as short as possible

## 18.5 UART BYPASS FUNCITON

When in UART bypass mode, UART2 is connect to USB interface; Otherwise, UART2 use normal UART interface.

Signal	CONNECT	I/O	Description
BYPASSDMDATA0	uart2_sout	I	Data for DM0 Transmitter Digital Bypass
BYPASSDMENO	uoc0_con0[8]	I	DM0 Transmitter Digital Bypass Enable
BYPASSSEL0	uoc0_con0[9]	I	Transmitter Digital Bypass mode Enable
FSVPLUS0	uart2_sin	O	Single-Ended D- Indicator The controller signal indicates the state of the DP during normal operation or UART data reception
OTGDISABLE0	uoc0_con0[4]	I	1'b1: OTG0 disable; 1'b0: OTG0 normal mode
COMMONONNN	uoc0_con0[0]	I	Common Block Power-Down Control This signal controls the power-down signals in PLL blocks when the USB PHY is in Suspend Mode. 1: PLL blocks are powered down. 0: PLL blocks remain powered This signal is a strapping option that must be set prior to a power-on reset and remain static during normal operation.

**Note: USB OTG2.0 PHY support UART Bypass Function.**

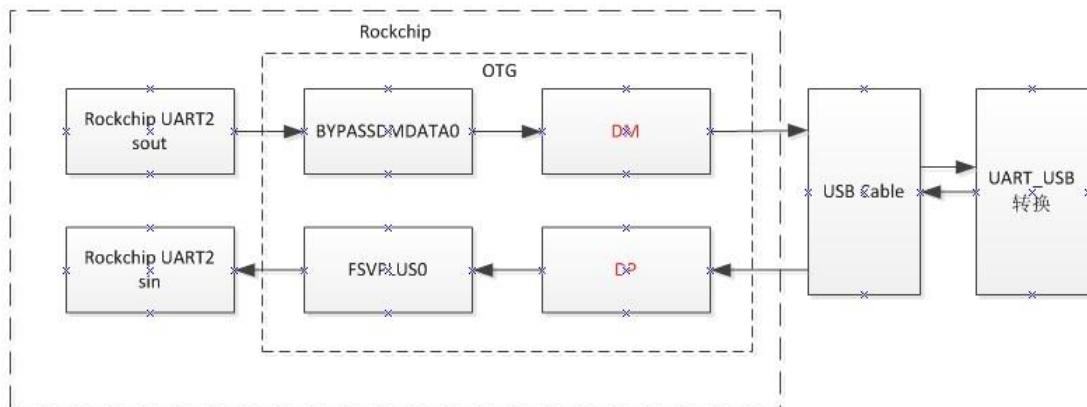


Fig. 18-10 UART Application

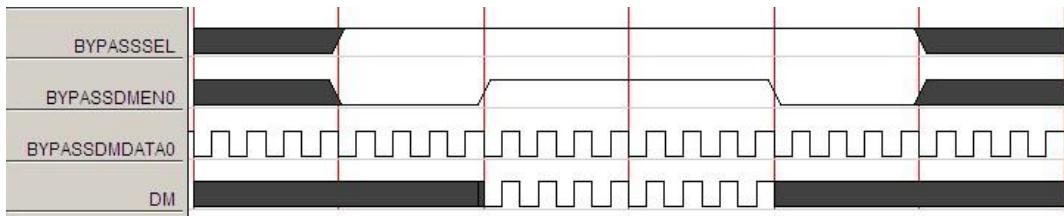


Fig. 18-11 UART Timing Sequence

To use UART and Auto resume functions:

1. Disable the OTG block by setting OTGDISABLE0 to 1'b1.
2. Disable the pull-up resistance on the D+ line by setting OPMODE0[1:0] to 2'b01.
3. To ensure that the XO, Bias, and PLL blocks are powered down in Suspend mode, set COMMONONNN to 1'b1.
4. Place the USB PHY in Suspend mode by setting SUSPENDM0 to 1'b0.
5. Set BYPASSEL0 to 1'b1.
6. To transmit data, controls BYPASSEMDENO, and BYPASSEMDATA0.

To receive data, monitor FSVPPLUS0.

To return to normal operating mode:

1. Ensure that there is no activity on the USB.
2. Set BYPASSEL0 to 1'b0.
3. setting SUSPENDM0 to 1'b1. Resume the USB PHY.
4. set COMMONONNN to 1'b0.
5. set OTGDISABLE0 to 1'b0.

## 18.6 Register Description

### 18.6.1 Register Summary

Name	Offset	Size	Reset Value	Description
------	--------	------	-------------	-------------

Name	Offset	Size	Reset Value	Description
USBOTG_GOTGCTL	0x0000	W	0x00000000	Control and Status Register
USBOTG_GOTGINT	0x0004	W	0x00000000	Interrupt Register
USBOTG_GAHBCFG	0x0008	W	0x00000000	AHB Configuration Register
USBOTG_GUSBCFG	0x000c	W	0x00001400	USB Configuration Register
USBOTG_GRSTCTL	0x0010	W	0x80000000	Reset Register
USBOTG_GINTSTS	0x0014	W	0x00000000	Interrupt Register
USBOTG_GINTMSK	0x0018	W	0x00000000	Interrupt Mask Register
USBOTG_GRXSTSR	0x001c	W	0x00000000	Receive Status Debug Read Register
USBOTG_GRXSTSP	0x0020	W	0x00000000	Receive Status Read and Pop Register
USBOTG_GRXFISZ	0x0024	W	0x00000000	Receive FIFO Size Register
USBOTG_GNPTXFSIZ	0x0028	W	0x00000000	Non-Periodic Transmit FIFO Size Register
USBOTG_GNPTXSTS	0x002c	W	0x00000000	Non-Periodic Transmit FIFO/Queue Status Register
USBOTG_GI2CCTL	0x0030	W	0x11000000	I2C Address Register
USBOTG_GPVNDCTL	0x0034	W	0x00000000	PHY Vendor Control Register
USBOTG_GGPIO	0x0038	W	0x00000000	General Purpose Input / Output Register
USBOTG_GUID	0x003c	W	0x00000000	User ID Register
USBOTG_GSNPSID	0x0040	W	0x00004f54	Core ID Register
USBOTG_GHWCFG1	0x0044	W	0x00000000	User HW Config1 Register
USBOTG_GHWCFG2	0x0048	W	0x00000000	User HW Config2 Register
USBOTG_GHWCFG3	0x004c	W	0x00000000	User HW Config3 Register
USBOTG_GHWCFG4	0x0050	W	0x00000000	User HW Config4 Register
USBOTG_GLPMCFG	0x0054	W	0x00000000	Core LPM Configuration Register
USBOTG_GPWRDN	0x0058	W	0x00000000	Global Power Down Register
USBOTG_GDFIFOFC G	0x005c	W	0x00000000	Global DFIFO Software Configuration Register
USBOTG_GADPCTL	0x0060	W	0x00000000	ADP Timer, Control and Status Register
USBOTG_HPTXFSIZ	0x0100	W	0x00000000	Host Periodic Transmit FIFO Size Register
USBOTG_DIEPTXF <sub>n</sub>	0x0104 +4*(n-1)	W	0x00000000	Device Periodic Transmit FIFO-n Size Register n = 1 - 15
USBOTG_HCFG	0x0400	W	0x00000000	Host Configuration Register
USBOTG_HFIR	0x0404	W	0x00000000	Host Frame Interval Register
USBOTG_HFNUM	0x0408	W	0x0000ffff	Host Frame Number/Frame Time Remaining Register

Name	Offset	Size	Reset Value	Description
USBOTG_HPTXSTS	0x0410	W	0x00000000	Host Periodic Transmit FIFO/Queue Status Register
USBOTG_HAINT	0x0414	W	0x00000000	Host All Channels Interrupt Register
USBOTG_HAINTMSK	0x0418	W	0x00000000	Host All Channels Interrupt Mask Register
USBOTG_HPRT	0x0440	W	0x00000000	Host Port Control and Status Register
USBOTG_HCCHARn	0x0500 +0x20 *n	W	0x00000000	Host Channel-n Characteristics Register n = 0 - 15
USBOTG_HCSPLTn	0x0504 +0x20 *n	W	0x00000000	Host Channel-n Split Control Register n = 0 - 15
USBOTG_HCINTn	0x0508 +0x20 *n	W	0x00000000	Host Channel-n Interrupt Register n = 0 - 15
USBOTG_HCINTMSKn	0x050c +0x20 *n	W	0x00000000	Host Channel-n Interrupt Mask Register n = 0 - 15
USBOTG_HCTSIZn	0x0510 +0x20 *n	W	0x00000000	Host Channel-n Transfer Size Register n = 0 - 15
USBOTG_HCDMAN	0x0514 +0x20 *n	W	0x00000000	Host Channel-n DMA Address Register n = 0 - 15
USBOTG_HCDMABn	0x051c +0x20 *n	W	0x00000000	Host Channel-n DMA Buffer Address Register n = 0 - 15
USBOTG_DCFG	0x0800	W	0x08200000	Device Configuration Register
USBOTG_DCTL	0x0804	W	0x00002000	Device Control Register
USBOTG_DSTS	0x0808	W	0x00000000	Device Status Register
USBOTG_DIEPMSK	0x0810	W	0x00000000	Device IN Endpoint common interrupt mask register
USBOTG_DOEPMSK	0x0814	W	0x00000000	Device OUT Endpoint common interrupt mask register
USBOTG_DAINT	0x0818	W	0x00000000	Device All Endpoints interrupt register
USBOTG_DAINTMSK	0x081c	W	0x00000000	Device All Endpoint interrupt mask register
USBOTG_DTKNQR1	0x0820	W	0x00000000	Device IN token sequence learning queue read register1

Name	Offset	Size	Reset Value	Description
USBOTG_DTKNQR2	0x0824	W	0x00000000	Device IN token sequence learning queue read register2
USBOTG_DVBUSDIS	0x0828	W	0x00000b8f	Device VBUS discharge time register
USBOTG_DVBUSPULSE	0x082c	W	0x00000000	Device VBUS Pulsing Timer Register
USBOTG_DTHRCTL	0x0830	W	0x08100020	Device Threshold Control Register
USBOTG_DIEPEMPMSK	0x0834	W	0x00000000	Device IN endpoint FIFO empty interrupt mask register
USBOTG_DEACHINT	0x0838	W	0x00000000	Device each endpoint interrupt register
USBOTG_DEACHINTMSK	0x083c	W	0x00000000	Device each endpoint interrupt register mask
USBOTG_DIEPEACHMSKn	0x0840 +4*n	W	0x00000000	Device each IN endpoint -n interrupt Register n = 0 - 15
USBOTG_DOEPEACHMSKn	0x0880 +4*n	W	0x00000000	Device each out endpoint-n interrupt register n = 0 - 15
USBOTG_DIEPCTL0	0x0900	W	0x00008000	Device control IN endpoint 0 control register
USBOTG_DIEPINTn	0x0908 +0x20 *n	W	0x00000000	Device Endpoint-n Interrupt Register n = 0 - 15
USBOTG_DIEPTSIZn	0x0910 +0x20 *n	W	0x00000000	Device endpoint n transfer size register n = 0 - 15
USBOTG_DIEPDMAAn	0x0914 +0x20 *n	W	0x00000000	Device endpoint-n DMA address register n = 0 - 15
USBOTG_DTXFSTS <sub>n</sub>	0x0918 +0x20 *n	W	0x00000000	Device IN endpoint transmit FIFO status register n = 0 - 15
USBOTG_DIEPDMAB <sub>n</sub>	0x091c	W	0x00000000	Device endpoint-n DMA buffer address register
USBOTG_DIEPCTL <sub>n</sub>	0x0920 +0x20 *(n-1)	W	0x00000000	Device endpoint-n control register n = 1 - 15
USBOTG_DOEPCTL0	0x0b00	W	0x00000000	Device control OUT endpoint 0 control register

Name	Offset	Size	Reset Value	Description
USBOTG_DOEPINTn	0x0b08 +0x20 *n	W	0x00000000	Device endpoint-n control register n = 0 - 15
USBOTG_DOEPTSIZn	0x0b10 +0x20 *n	W	0x00000000	Device endpoint n transfer size register n = 0 - 15
USBOTG_DOEPDMAn	0x0b14 +0x20 *n	W	0x00000000	Device Endpoint-n DMA Address Register n = 0 - 15
USBOTG_DOEPDMABn	0x0b1c +0x20 *n	W	0x00000000	Device endpoint-n DMA buffer address register n = 0 - 15
USBOTG_DOEPCTLn	0x0b20 +0x20 *(n-1)	W	0x00000000	Device endpoint-n control register n = 1 - 15
USBOTG_PCGCR	0x0b24	W	0x200b8000	Power and clock gating control register

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

### 18.6.2 Detail Register Description

#### USBOTG\_GOTGCTL

Address: Operational Base + offset (0x0000)

Control and Status Register

Bit	Attr	Reset Value	Description
31:28	RO	0x0	reserved
27	RW	0x0	ChirpEn Chirp on enable This bit when programmed to 1'b1 results in the core asserting chirp_on before sending an actual Chirp "K" signal on USB. This bit is present only if OTG_BC_SUPPORT = 1. If OTG_BC_SUPPORT != 1, this bit is a reserved bit.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
26:22	RO	0x00	MultValidBc Multi Valued ID pin Battery Charger ACA inputs in the following order: Bit 26 - rid_float. Bit 25 - rid_gnd Bit 24 - rid_a Bit 23 - rid_b Bit 22 - rid_c These bits are present only if OTG_BC_SUPPORT = 1. Otherwise, these bits are reserved and will read 5'h0.
21	RO	0x0	reserved
20	RW	0x0	OTGVer OTG version Indicates the OTG revision. 1'b0: OTG Version 1.3. In this version the core supports Data line pulsing and VBus pulsing for SRP. 1'b1: OTG Version 2.0. In this version the core supports only Data line pulsing for SRP.
19	RO	0x0	BSesVld B-session valid Indicates the Device mode transceiver status. 1'b0: B-session is not valid. 1'b1: B-session is valid. In OTG mode, you can use this bit to determine if the device is connected or disconnected. Note: If you do not enable OTG features (such as SRP and HNP), the read reset value will be 1. The vbus assigns the values internally for non-SRP or non-HNP configurations.
18	RO	0x0	ASesVld A-session valid Indicates the Host mode transceiver status. 1'b0: A-session is not valid 1'b1: A-session is valid Note: If you do not enable OTG features (such as SRP and HNP), the read reset value will be 1. The vbus assigns the values internally for non-SRP or non-HNP configurations.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17	RO	0x0	DbnTime Long/short debounce time Indicates the debounce time of a detected connection. 1'b0: Long debounce time, used for physical connections (100 ms + 2.5 us) 1'b1: Short debounce time, used for soft connections (2.5 us)
16	RO	0x0	ConIDSsts Connector ID Status Indicates the connector ID status on a connect event. 1'b0: The core is in A-Device mode 1'b1: The core is in B-Device mode
15:12	RO	0x0	reserved
11	RW	0x0	DevHNPEn Device HNP Enable The application sets this bit when it successfully receives a SetFeature. SetHNPEnable command from the connected USB host. 1'b0: HNP is not enabled in the application 1'b1: HNP is enabled in the application
10	RW	0x0	HstSetHNPEn Host set HNP enable The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device. 1'b0: Host Set HNP is not enabled 1'b1: Host Set HNP is enabled
9	RW	0x0	HNPReq HNP request The application sets this bit to initiate an HNP request to the connected USB host. The application can clear this bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (GOTGINT.HstNegSucStsChng) is set. The core clears this bit when the HstNegSucStsChng bit is cleared. 1'b0: No HNP request 1'b1: HNP request

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	RO	0x0	HstNegScs Host Negotiation Success The core sets this bit when host negotiation is successful. The core clears this bit when the HNP Request (HNPReq) bit in this register is set. 1'b0: Host negotiation failure 1'b1: Host negotiation success
7:2	RO	0x0	reserved
1	RW	0x0	SesReq Session Request The application sets this bit to initiate a session request on the USB. The application can clear this bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (GOTGINT.HstNegSucStsChng) is set. The core clears this bit when the HstNegSucStsChng bit is cleared. If you use the USB 1.1 Full-Speed Serial Transceiver interface to initiate the session request, the application must wait until the VBUS discharges to 0.2 V, after the B-Session Valid bit in this register (GOTGCTL.BSesVld) is cleared. This discharge time varies between different PHYs and can be obtained from the PHY vendor. 1'b0: No session request 1'b1: Session request
0	RO	0x0	SesReqScs Session Request Success The core sets this bit when a session request initiation is successful. 1'b0: Session request failure 1'b1: Session request success

**USBOTG\_GOTGINT**

Address: Operational Base + offset (0x0004)

Interrupt Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:21	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
20	W1C	0x0	MultiValueChg Multi-Valued input changed This bit when set indicates that there is a change in the value of at least one ACA pin value. This bit is present only if OTG_BC_SUPPORT = 1, otherwise it is reserved.
19	W1C	0x0	DbnceDone Debounce Done The core sets this bit when the debounce is completed after the device connection. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable or SRP Capable bit is set in the Core USB Configuration register (GUSBCFG.HNPCap or GUSBCFG.SRPCap, respectively).
18	W1C	0x0	ADevTOUTChg A-Device Timeout Change The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.
17	W1C	0x0	HstNegDet Host Negotiation Detected The core sets this bit when it detects a host negotiation request on the USB
16:10	RO	0x0	reserved
9	W1C	0x0	HstNegSucStsChng Host Negotiation Success Status Change The core sets this bit on the success or failure of a USB host negotiation request. The application must read the Host Negotiation Success bit of the OTG Control and Status register (GOTGCTL.HstNegScs) to check for success or failure
8	W1C	0x0	SesReqSucStsChng Session Request Success Status Change The core sets this bit on the success or failure of a session request. The application must read the Session Request Success bit in the OTG Control and Status register (GOTGCTL.SesReqScs) to check for success or failure.
7:3	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	W1C	0x0	SesEndDet Session End Detected The core sets this bit when the utmisrp_bvalid signal is deasserted
1:0	RO	0x0	reserved

**USBOTG\_GAHBCFG**

Address: Operational Base + offset (0x0008)

AHB Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:23	RO	0x0	reserved
22	RW	0x0	NotiAllDmaWrit Notify All Dma Write Transactions This bit is programmed to enable the System DMA Done functionality for all the DMA write Transactions corresponding to the Channel/Endpoint. This bit is valid only when GAHBCFG.RemMemSupp is set to 1. GAHBCFG.NotiAllDmaWrit = 1. HSOTG core asserts int_dma_req for all the DMA write transactions on the AHB interface along with int_dma_done, chep_last_transact and chep_number signal informations. The core waits for sys_dma_done signal for all the DMA write transactions in order to complete the transfer of a particular Channel/Endpoint. GAHBCFG.NotiAllDmaWrit = 0. HSOTG core asserts int_dma_req signal only for the last transaction of DMA write transfer corresponding to a particular Channel/Endpoint. Similarly, the core waits for sys_dma_done signal only for that transaction of DMA write to complete the transfer of a particular Channel/Endpoint.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
21	RW	0x0	<p>RemMemSupp Remote Memory Support This bit is programmed to enable the functionality to wait for the system DMA Done Signal for the DMA Write Transfers. GAHBCFG.RemMemSupp=1.</p> <p>The int_dma_req output signal is asserted when HSOTG DMA starts write transfer to the external memory. When the core is done with the Transfers it asserts int_dma_done signal to flag the completion of DMA writes from HSOTG. The core then waits for sys_dma_done signal from the system to proceed further and complete the Data Transfer corresponding to a particular Channel/Endpoint.</p> <p>GAHBCFG.RemMemSupp=0.</p> <p>The int_dma_req and int_dma_done signals are not asserted and the core proceeds with the assertion of the XferComp interrupt as soon as the DMA write transfer is done at the HSOTG Core Boundary and it does not wait for the sys_dma_done signal to complete the DATA transfers.</p>
20:9	RO	0x0	reserved
8	RW	0x0	<p>PTxFEmpLvl Periodic TxFIFO Empty Level Indicates when the Periodic TxFIFO Empty Interrupt bit in the Core Interrupt register (GINTSTS.PTxFEmp) is triggered. This bit is used only in Slave mode.</p> <p>1'b0: GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is half empty 1'b1: GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is completely empty</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	<p>NPTxFEmpLvl Non-Periodic TxFIFO Empty Level This bit is used only in Slave mode. In host mode and with Shared FIFO with device mode, this bit indicates when the Non-Periodic TxFIFO Empty Interrupt bit in the Core Interrupt register GINTSTS.NPTxFEmp) is triggered. With dedicated FIFO in device mode, this bit indicates when IN endpoint Transmit FIFO empty interrupt (DIEPINTn.TxFEmp) is triggered.</p> <p>Host mode and with Shared FIFO with device mode:</p> <p>1'b0: GINTSTS.NPTxFEmp interrupt indicates that the Non-Periodic TxFIFO is half empty 1'b1: GINTSTS.NPTxFEmp interrupt indicates that the Non-Periodic TxFIFO is completely empty</p> <p>Dedicated FIFO in device mode:</p> <p>1'b0: DIEPINTn.TxFEmp interrupt indicates that the IN Endpoint TxFIFO is half empty 1'b1: DIEPINTn.TxFEmp interrupt indicates that the IN Endpoint TxFIFO is completely empty</p>
6	RO	0x0	reserved
5	RW	0x0	<p>DMAEn DMA Enable 1'b0: Core operates in Slave mode 1'b1: Core operates in a DMA mode This bit is always 0 when Slave-Only mode has been selected.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:1	RW	0x0	<p>HBstLen Burst Length/Type This field is used in both External and Internal DMA modes. In External DMA mode, these bits appear on dma_burst[3:0] ports, External DMA Mode defines the DMA burst length in terms of 32-bit words:</p> <p>4'b0000: 1 word 4'b0001: 4 words 4'b0010: 8 words 4'b0011: 16 words 4'b0100: 32 words 4'b0101: 64 words 4'b0110: 128 words 4'b0111: 256 words Others: Reserved</p> <p>Internal DMA Mode AHB Master burst type:</p> <p>4'b0000: Single 4'b0001: INCR 4'b0011: INCR4 4'b0101: INCR8 4'b0111: INCR16 Others: Reserved</p>
0	RW	0x0	<p>GlblIntrMsk Global Interrupt Mask The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit's setting, the interrupt status registers are updated by the core.</p> <p>1'b0: Mask the interrupt assertion to the application. 1'b1: Unmask the interrupt assertion to the application.</p>

**USBOTG\_GUSBCFG**

Address: Operational Base + offset (0x000c)

USB Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	<p>CorruptTxpacket Corrupt Tx packet This bit is for debug purposes only. Never set this bit to 1.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
30	RW	0x0	<p>ForceDevMode Force Device Mode Writing a 1 to this bit forces the core to device mode irrespective of utmiotg_iddig input pin. 1'b0: Normal Mode 1'b1: Force Device Mode After setting the force bit, the application must wait at least 25 ms before the change to take effect. When the simulation is in scale down mode, waiting for 500 us is sufficient. This bit is valid only when OTG_MODE = 0, 1 or 2. In all other cases, this bit reads 0.</p>
29	RW	0x0	<p>ForceHstMode Force Host Mode Writing a 1 to this bit forces the core to host mode irrespective of utmiotg_iddig input pin. 1'b0: Normal Mode 1'b1: Force Host Mode After setting the force bit, the application must wait at least 25 ms before the change to take effect. When the simulation is in scale down mode, waiting for 500 us is sufficient. This bit is valid only when OTG_MODE =0, 1 or 2. In all other cases, this bit reads 0.</p>
28	RW	0x0	<p>TxEndDelay Tx End Delay Writing a 1 to this bit enables the TxEndDelay timers in the core. 1'b0: Normal mode 1'b1: Introduce Tx end delay timers</p>
27	RW	0x0	<p>IC_USB_TrafficCtl IC_USB Traffic Pull Remove Control When this bit is set, pullup/pulldown resistors are detached from the USB during traffic signaling. This bit is valid only when configuration parameter OTG_ENABLE_IC_USB = 1 and register field GUSBCFG.IC_USBCap is set to 1.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
26	RW	0x0	<p>IC_USBCap IC_USB-Capable The application uses this bit to control the IC_USB capabilities. 1'b0: IC_USB PHY Interface is not selected. 1'b1: IC_USB PHY Interface is selected.</p> <p>This bit is writable only if OTG_ENABLE_IC_USB=1 and OTG_FSPHY_INTERFACE!=0. The reset value depends on the configuration parameter OTG_SELECT_IC_USB when OTG_ENABLE_IC_USB = 1. In all other cases, this bit is set to 1'b0 and the bit is read only.</p>
25	RW	0x0	<p>ULPIIfDis ULPI Interface Protect Disable Controls circuitry built into the PHY for protecting the ULPI interface when the link tri-states STP and data. Any pull-ups or pull-downs employed by this feature can be disabled. Please refer to the ULPI Specification for more detail.</p> <p>1'b0: Enables the interface protect circuit 1'b1: Disables the interface protect circuit</p>
24	RW	0x0	<p>IndPassThrough Indicator Pass Through Controls whether the Complement Output is qualified with the Internal Vbus Valid comparator before being used in the Vbus State in the RX CMD. Please refer to the ULPI Specification for more detail.</p> <p>1'b0: Complement Output signal is qualified with the Internal VbusValid comparator. 1'b1: Complement Output signal is not qualified with the Internal VbusValid comparator.</p>
23	RW	0x0	<p>IndComple Indicator Complement Controls the PHY to invert the ExternalVbusIndicator input signal, generating the Complement Output. Please refer to the ULPI Specification for more detail</p> <p>1'b0: PHY does not invert External Vbus Indicator signal 1'b1: PHY does invert External Vbus Indicator signal</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
22	RW	0x0	<p>TermSelDLPulse TermSel DLine Pulsing Selection This bit selects utmi_termselect to drive data line pulse during SRP.</p> <p>1'b0: Data line pulsing using utmi_txvalid (default).</p> <p>1'b1: Data line pulsing using utmi_termsel.</p>
21	RW	0x0	<p>ULPIExtVbusIndicator ULPI External VBUS Indicator This bit indicates to the ULPI PHY to use an external VBUS over-current indicator.</p> <p>1'b0: PHY uses internal VBUS valid comparator.</p> <p>1'b1: PHY uses external VBUS valid comparator. (Valid only when RTL parameter OTG_HSPHY_INTERFACE = 2 or 3)</p>
20	RW	0x0	<p>ULPIExtVbusDrv ULPI External VBUS Drive This bit selects between internal or external supply to drive 5V on VBUS,in ULPI PHY.</p> <p>1'b0: PHY drives VBUS using internal charge pump (default).</p> <p>1'b1: PHY drives VBUS using external supply. (Valid only when RTL parameter OTG_HSPHY_INTERFACE = 2 or 3)</p>
19	RW	0x0	<p>ULPIClkSusM ULPI Clock SuspendM This bit sets the ClockSuspendM bit in the Interface Control register on the ULPI PHY. This bit applies only in serial or carkit modes.</p> <p>1'b0: PHY powers down internal clock during suspend.</p> <p>1'b1: PHY does not power down internal clock. (Valid only when RTL parameter OTG_HSPHY_INTERFACE = 2 or 3)</p>
18	RW	0x0	<p>ULPIAutoRes ULPI Auto Resume This bit sets the AutoResume bit in the Interface Control register on the ULPI PHY.</p> <p>1'b0: PHY does not use AutoResume feature.</p> <p>1'b1: PHY uses AutoResume feature. (Valid only when RTL parameter OTG_HSPHY_INTERFACE = 2 or 3)</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17	RW	0x0	<p>ULPIFsLs ULPI FS/LS Select The application uses this bit to select the FS/LS serial interface for the ULPI PHY. This bit is valid only when the FS serial transceiver is selected on the ULPI PHY.</p> <p>1'b0: ULPI interface 1'b1: ULPI FS/LS serial interface (Valid only when RTL parameters OTG_HSPHY_INTERFACE = 2 or 3 and OTG_FSPHY_INTERFACE = 1, 2, or 3)</p>
16	RW	0x0	<p>OtgI2CSel UTMIFS or I2C Interface Select The application uses this bit to select the I2C interface.</p> <p>1'b0: UTMI USB 1.1 Full-Speed interface for OTG signals 1'b1: I2C interface for OTG signals This bit is writable only if I2C and UTMIFS were specified for Enable I2C Interface? (parameter OTG_I2C_INTERFACE = 2). Otherwise, reads return 0.</p>
15	RW	0x0	<p>PhyLPwrClkSel PHY Low-Power Clock Select Selects either 480-MHz or 48-MHz (low-power) PHY mode. In FS and LS modes, the PHY can usually operate on a 48-MHz clock to save power.</p> <p>1'b0: 480-MHz Internal PLL clock 1'b1: 48-MHz External Clock In 480 MHz mode, the UTMI interface operates at either 60 or 30-MHz, depending upon whether 8- or 16-bit data width is selected. In 48-MHz mode, the UTMI interface operates at 48 MHz in FS and LS modes. This bit drives the utmi_fsls_low_power core output signal, and is valid only for UTMI+ PHYs.</p>
14	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13:10	RW	0x5	<p>USBTrdTim USB Turnaround Time Sets the turnaround time in PHY clocks. Specifies the response time for a MAC request to the Packet FIFO Controller (PFC) to fetch data from the DFIF(SPRAM). This must be programmed to</p> <p>4'h5: When the MAC interface is 16-bit UTMI+. 4'h9: When the MAC interface is 8-bit UTMI+. Note: The values above are calculated for the minimum AHB frequency of 30 MHz. USB turnaround time is critical for certification where long cables and 5-Hubs are used, so if you need the AHB to run at less than 30 MHz, and if USB turnaround time is not critical, these bits can be programmed to a larger value.</p>
9	RW	0x0	<p>HNPCap HNP-Capable The application uses this bit to control the Otg core's HNP capabilities.</p> <p>1'b0: HNP capability is not enabled. 1'b1: HNP capability is enabled. This bit is writable only if an HNP mode was specified for Mode of Operation (parameter OTG_MODE). Otherwise, reads return 0.</p>
8	RW	0x0	<p>SRPCap SRP-Capable The application uses this bit to control the Otg core SRP capabilities. If the core operates as a non-SRP-capable B-device, it cannot request the connected A-device (host) to activate VBUS and start a session.</p> <p>0: SRP capability is not enabled. 1: SRP capability is enabled. This bit is writable only if an SRP mode was specified for Mode of Operation (parameter OTG_MODE). Otherwise, reads return 0.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	<p>DDRSel ULPI DDR Select The application uses this bit to select a Single Data Rate (SDR) or Double Data Rate (DDR) or ULPI interface.</p> <p>1'b0: Single Data Rate ULPI Interface, with 8-bit-wide data bus 1'b1: Double Data Rate ULPI Interface, with 4-bit-wide data bus</p>
6	RW	0x0	<p>PHYSel USB 2.0 High-Speed PHY or USB 1.1 Full-Speed Serial Transceiver The application uses this bit to select either a high-speed UTMI+ or ULPI PHY, or a full-speed transceiver.</p> <p>1'b0: USB 2.0 high-speed UTMI+ or ULPI PHY 1'b1: USB 1.1 full-speed serial transceiver If a USB 1.1 Full-Speed Serial Transceiver interface was not selected (parameter OTG_FSPHY_INTERFACE = 0), this bit is always 0, with Write Only access. If a high-speed PHY interface was not selected (parameter OTG_HSPHY_INTERFACE = 0), this bit is always 1, with Write Only access. If both interface types were selected (parameters have non-zero values), the application uses this bit to select which interface is active, and access is Read and Write.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	<p>FSIntf Full-Speed Serial Interface Select The application uses this bit to select either a unidirectional or bidirectional USB 1.1 full-speed serial transceiver interface. 1'b0: 6-pin unidirectional full-speed serial interface 1'b1: 3-pin bidirectional full-speed serial interface If a USB 1.1 Full-Speed Serial Transceiver interface was not selected (parameter OTG_FSPHY_INTERFACE = 0), this bit is always 0, with Write Only access. If a USB 1.1 FS interface was selected (parameter OTG_FSPHY_INTERFACE! = 0), then the application can set this bit to select between the 3- and 6-pin interfaces, and access is Read and Write.</p>
4	RW	0x0	<p>ULPI_UTMI_Sel ULPI or UTMI+ Select The application uses this bit to select either a UTMI+ interface or ULPI Interface. 1'b0: UTMI+ Interface 1'b1: ULPI Interface This bit is writable only if UTMI+ and ULPI was specified for High-Speed PHY Interface(s) (parameter OTG_HSPHY_INTERFACE = 3). Otherwise, reads return either 0 or 1, depending on the interface selected using the OTG_HSPHY_INTERFACE parameter.</p>
3	RW	0x0	<p>PHYIf PHY Interface The application uses this bit to configure the core to support a UTMI+ PHY with an 8- or 16-bit interface. When a ULPI PHY is chosen, this must be set to 8-bit mode. 1'b0: 8 bits 1'b1: 16 bits This bit is writable only if UTMI+ and ULPI were selected (parameter OTG_HSPHY_DWIDTH = 3). Otherwise, this bit returns the value for the power-on interface selected during configuration.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:0	RW	0x0	<p>TOutCal HS/FS Timeout Calibration The number of PHY clocks that the application programs in this field is added to the high-speed/full-speed inter-packet timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.</p> <p>The USB standard timeout value for high-speed operation is 736 to 816 (inclusive) bit times. The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock are:</p> <p>High-speed operation: One 30-MHz PHY clock = 16 bit times One 60-MHz PHY clock = 8 bit times</p> <p>Full-speed operation: One 30-MHz PHY clock = 0.4 bit times One 60-MHz PHY clock = 0.2 bit times One 48-MHz PHY clock = 0.25 bit times</p>

**USBOTG\_GRSTCTL**

Address: Operational Base + offset (0x0010)

Reset Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x1	AHBIdle AHB Master Idle Indicates that the AHB Master State Machine is in the IDLE condition.
30	RO	0x0	DMAReq DMA Request Signal Indicates that the DMA request is in progress. Used for debug.
29:11	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
10:6	RW	0x00	<p>TxFNum Tx FIFO Number This is the FIFO number that must be flushed using the Tx FIFO Flush bit. This field must not be changed until the core clears the Tx FIFO Flush bit.</p> <p>5'h0: Non-periodic Tx FIFO flush in Host mode; Non-periodic Tx FIFO flush in device mode when in shared FIFO operation. Tx FIFO 0 flush in device mode when in dedicated FIFO mode.</p> <p>5'h1: Periodic Tx FIFO flush in Host mode: Periodic Tx FIFO 1 flush in Device mode when in shared FIFO operation; TX FIFO 1 flush in device mode when in dedicated FIFO mode.</p> <p>5'h2: Periodic Tx FIFO 2 flush in Device mode when in shared FIFO operation: TX FIFO 2 flush in device mode when in dedicated FIFO mode.</p> <p>...</p> <p>5'hF: Periodic Tx FIFO 15 flush in Device mode when in shared FIFO operation: TX FIFO 15 flush in device mode when in dedicated FIFO mode.</p> <p>5'h10: Flush all the transmit FIFOs in device or host mode.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RWSC	0x0	<p>TxFFlsh Tx FIFO Flush</p> <p>This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction. The application must write this bit only after checking that the core is neither writing to the Tx FIFO nor reading from the Tx FIFO. Verify using these registers: Read NAK Effective Interrupt ensures the core is not reading from the FIFO. Write GRSTCTL.AHBIdle ensures the core is not writing anything to the FIFO.</p> <p>Flushing is normally recommended when FIFOs are re-configured or when switching between Shared FIFO and Dedicated Transmit FIFO operation. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.</p>
4	RWSC	0x0	<p>RxFFlsh Rx FIFO Flush</p> <p>The application can flush the entire Rx FIFO using this bit, but must first ensure that the core is not in the middle of a transaction. The application must only write to this bit after checking that the core is neither reading from the Rx FIFO nor writing to the Rx FIFO. The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.</p>
3	RWSC	0x0	<p>INTknQFlsh IN Token Sequence Learning Queue Flush</p> <p>This bit is valid only if OTG_EN_DED_TX_FIFO = 0. The application writes this bit to flush the IN Token Sequence Learning Queue.</p>
2	W1C	0x0	<p>FrmCntrRst Host Frame Counter Reset</p> <p>The application writes this bit to reset the (micro) frame number counter inside the core. When the (micro) frame counter is reset, the subsequent SOF sent out by the core has a (micro) frame number of 0.</p>

Bit	Attr	Reset Value	Description
1	RWSC	0x0	Reset A write to this bit issues a soft reset to the Otg_power_dn module of the core.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RWSC	0x0	<p>CSftRst Core Soft Reset Resets the hclk and phy_clock domains as follows: Clears the interrupts and all the CSR registers except the following register bits:</p> <ul style="list-style-type: none"> <li>PCGCCTL.RstPdwnModule</li> <li>PCGCCTL.GateHclk</li> <li>PCGCCTL.PwrClmp</li> <li>PCGCCTL.StopPPhyLPwrClkSelClk</li> <li>GUSBCFG.PhyLPwrClkSel</li> <li>GUSBCFG.DDRSel</li> <li>GUSBCFG.PHYSel</li> <li>GUSBCFG.FSIntf</li> <li>GUSBCFG.ULPI_UTMI_Sel</li> <li>GUSBCFG.PHYIf</li> <li>HCFG.FSLSPclkSel</li> <li>DCFG.DevSpd</li> <li>GGPIO</li> <li>GPWRDN</li> <li>GADPCTL</li> </ul> <p>All module state machines (except the AHB Slave Unit) are reset to the IDLE state, and all the transmit FIFOs and the receive FIFO are flushed. Any transactions on the AHB Master are terminated as soon as possible, after gracefully completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately. When Hibernation or ADP feature is enabled, the PMU module is not reset by the Core Soft Reset. The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit is cleared software must wait at least 3 PHY clocks before doing any access to the PHY domain (synchronization delay). Software must also must check that bit 31 of this register is 1 (AHB Master is IDLE) before starting any operation. Typically software reset is used during software development and also when you dynamically change the PHY selection bits in the USB configuration registers listed above. When you change the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.</p>

**USBOTG\_GINTSTS**

Address: Operational Base + offset (0x0014)

Interrupt Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	W1C	0x0	<p>WkUpInt Resume/Remote Wakeup Detected Interrupt Wakeup Interrupt during Suspend (L2) or LPM(L1) state.</p> <p>During Suspend(L2): Device Mode: This interrupt is asserted only when Host Initiated Resume is detected on USB. Host Mode: This interrupt is asserted only when Device Initiated Remote Wakeup is detected on USB.</p> <p>During LPM(L1): Device Mode: This interrupt is asserted for either Host Initiated Resume or Device Initiated Remote Wakeup on USB. Host Mode: This interrupt is asserted for either Host Initiated Resume or Device Initiated Remote Wakeup on USB.</p>
30	W1C	0x0	<p>SessReqInt Session Request/New Session Detected Interrupt</p> <p>In Host mode, this interrupt is asserted when a session request is detected from the device. In Host mode, this interrupt is asserted when a session request is detected from the device. In Device mode, this interrupt is asserted when the utmisrp_bvalid signal goes high.</p>
29	W1C	0x0	<p>Disconnect Disconnect Detected Interrupt</p> <p>This interrupt is asserted when a device disconnect is detected.</p>
28	W1C	0x0	<p>ConIDStsChng Connector ID Status Change</p> <p>This interrupt is asserted when there is a change in connector ID status.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27	W1C	0x0	LPM_Int LPM Transaction Received Interrupt Device Mode : This interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response. Host Mode : This interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (GLPMCFG.RetryCnt). This field is valid only if the Core LPM Configuration register's LPMCapable (LPMCap) field is set to 1.
26	RO	0x0	PTxFEmp Periodic TxFIFO Empty This interrupt is asserted when the Periodic Transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the Periodic Request Queue. The half or completely empty status is determined by the Periodic TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.PTxFEmpLvl).
25	RO	0x0	HChInt Host Channels Interrupt The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in Host mode). The application must read the Host All Channels Interrupt (HAINT) register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding Host Channel-n Interrupt (HCINTn) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the HCINTn register to clear this bit.
24	RO	0x0	PrtInt Host Port Interrupt The core sets this bit to indicate a change in port status of one of the OTG core ports in Host mode. The application must read the Host Port Control and Status (HPRT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the Host Port Control and Status register to clear this bit.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23	RW	0x0	ResetDet Reset Detected Interrupt The core asserts this interrupt in Device mode when it detects a reset on the USB in Partial Power-Down mode when the device is in Suspend. This interrupt is not asserted in Host mode.
22	W1C	0x0	FetSusp Data Fetch Suspended This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or Request Queue space. This interrupt is used by the application for an endpoint mismatch algorithm.
21	W1C	0x0	incomplP Incomplete Periodic Transfer In Host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending which are scheduled for the current micro-frame. Incomplete Isochronous OUT Transfer (incompISOOUT) The Device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current micro-frame. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register.
20	W1C	0x0	incompISOIN Incomplete Isochronous IN Transfer The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current micro-frame. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register. Note: This interrupt is not asserted in Scatter/Gather DMA mode.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19	RO	0x0	OEPInt OUT Endpoints Interrupt The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (DAINT) register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding Device OUT Endpoint-n Interrupt (DOEPINTn) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding DOEPINTn register to clear this bit.
18	RO	0x0	IEPInt IN Endpoints Interrupt The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (DAINT) register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding Device IN Endpoint-n Interrupt (DIEPINTn) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding DIEPINTn register to clear this bit.
17	W1C	0x0	EPMIs Endpoint Mismatch Interrupt Note: This interrupt is valid only in shared FIFO operation. Indicates that an IN token has been received for a non-periodic endpoint, but the data for another endpoint is present in the top of the Non-periodic Transmit FIFO and the IN endpoint mismatch count programmed by the application has expired.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16	W1C	0x0	RstrDoneInt Restore Done Interrupt The core sets this bit to indicate that the restore command after Hibernation was completed by the core. The core continues from Suspended state into the mode dictated by PCGCCTL.RestoreMode field. This bit is valid only when Hibernation feature is enabled.
15	W1C	0x0	EOPF End of Periodic Frame Interrupt Indicates that the period specified in the Periodic Frame Interval field of the Device Configuration register (DCFG.PerFrInt) has been reached in the current microframe.
14	W1C	0x0	ISOOutDrop Isochronous OUT Packet Dropped Interrupt The core sets this bit when it fails to write an isochronous OUT packet into the RxFIFO because the RxFIFO does not have enough space to accommodate a maximum packet size packet for the isochronous OUT endpoint.
13	W1C	0x0	EnumDone Enumeration Done The core sets this bit to indicate that speed enumeration is complete. The application must read the Device Status (DSTS) register to obtain the enumerated speed.
12	W1C	0x0	USBRst USB Reset The core sets this bit to indicate that a reset is detected on the USB.
11	W1C	0x0	USBSusp USB Suspend The core sets this bit to indicate that a suspend was detected on the USB. The core enters the Suspended state when there is no activity on the utmi_linestate signal for an extended period of time.
10	W1C	0x0	ErlySusp Early Suspend The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	W1C	0x0	I2CINT I2C Interrupt The core sets this interrupt when I2C access is completed on the I2C interface. This field is used only if the I2C interface was enabled . Otherwise, reads return 0.
8	W1C	0x0	ULPICKINT ULPI Carkit Interrupt This field is used only if the Carkit interface was enabled . Otherwise, reads return 0. The core sets this interrupt when a ULPI Carkit interrupt is received. The core's PHY sets ULPI Carkit interrupt in UART or Audio mode. I2C Carkit Interrupt (I2CCKINT) This field is used only if the I2C interface was enabled . Otherwise, reads return 0.The core sets this interrupt when a Carkit interrupt is received. The core's PHY sets the I2C Carkit interrupt in Audio mode.
7	RO	0x0	GOUTNakEff Global OUT NAK Effective Indicates that the Set Global OUT NAK bit in the Device Control register DCTL.SGOUTNak), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear Global OUT NAK bit in the Device Control register (DCTL.CGOUTNak).
6	RO	0x0	GINNakEff Global IN Non-Periodic NAK Effective Indicates that the Set Global Non-periodic IN NAK bit in the Device Control register (DCTL.SGNPInNak), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear Global Nonperiodic IN NAK bit in the Device Control register (DCTL.CGNPInNak). This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RO	0x0	NPTxFEmp Non-Periodic TxFIFO Empty This interrupt is valid only when OTG_ENDED_TX_FIFO = 0. This interrupt is asserted when the Non-periodic TxFIFO is either half or completely empty, and there is space for at least one entry to be written to the Non-periodic Transmit Request Queue. The half or completely empty status is determined by the Non-periodic TxFIFO Empty Level bit in the Core AHB Configuration register(GAHBCFG.NPTxFEmpLvl).
4	RO	0x0	RxFLvl RxFIFO Non-Empty Indicates that there is at least one packet pending to be read from the RxFIFO.
3	W1C	0x0	Sof Start of (micro)Frame In Host mode, the core sets this bit to indicate that an SOF (FS), micro-SOF(HS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt. In Device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the Device Status register to get the current (micro) frame number. This interrupt is seen only when the core is operating at either HS or FS.
2	RO	0x0	OTGInt OTG Interrupt The core sets this bit to indicate an OTG protocol event. The application must read the OTG Interrupt Status (GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the GOTGINT register to clear this bit.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	W1C	0x0	ModeMis Mode Mismatch Interrupt The core sets this bit when the application is trying to access: A Host mode register, when the core is operating in Device mode; A Device mode register, when the core is operating in Host mode. The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.
0	RO	0x0	CurMod Current Mode of Operation Indicates the current mode. 1'b0: Device mode 1'b1: Host mode

**USBOTG\_GINTMSK**

Address: Operational Base + offset (0x0018)

Interrupt Mask Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	WkUpIntMsk Resume/Remote Wakeup Detected Interrupt Mask
30	RW	0x0	SessReqIntMsk Session Request/New Session Detected Interrupt Mask
29	RW	0x0	DisconnIntMsk Disconnect Detected Interrupt Mask
28	RW	0x0	ConIDStsChngMsk Connector ID Status Change Mask
27	RW	0x0	LPM_IntMsk LPM Transaction Received Interrupt Mask
26	RW	0x0	PTxFEmpMsk Periodic TxFIFO Empty Mask
25	RW	0x0	HChIntMsk Host Channels Interrupt Mask
24	RW	0x0	PrtIntMsk Host Port Interrupt Mask
23	RW	0x0	ResetDetMsk Reset Detected Interrupt Mask
22	RW	0x0	FetSuspMsk Data Fetch Suspended Mask

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
21	RW	0x0	incomlPMsk_incompISOOUTMsk Incomplete Periodic Transfer Mask(Host only) Incomplete Isochronous OUT Transfer Mask(Device only)
20	RW	0x0	incompISOINMsk Incomplete Isochronous IN Transfer Mask
19	RW	0x0	OEPIntMsk OUT Endpoints Interrupt Mask
18	RW	0x0	IEPIntMsk IN Endpoints Interrupt Mask
17	RW	0x0	EPMisMsk Endpoint Mismatch Interrupt Mask
16	RW	0x0	RstrDoneIntMsk Restore Done Interrupt Mask This field is valid only when Hibernation feature is enabled.
15	RW	0x0	EOPFMsk End of Periodic Frame Interrupt Mask
14	RW	0x0	ISOOutDropMsk Isochronous OUT Packet Dropped Interrupt Mask
13	RW	0x0	EnumDoneMsk Enumeration Done Mask
12	RW	0x0	USBRstMsk USB Reset Mask
11	RW	0x0	USBSuspMsk USB Suspend Mask
10	RW	0x0	ErlySuspMsk Early Suspend Mask
9	RW	0x0	I2CIntMsk I2C Interrupt Mask
8	RW	0x0	ULPICKINTMsk_I2CCKINTMsk ULPI Carkit Interrupt Mask (ULPICKINTMsk) I2C Carkit Interrupt Mask (I2CCKINTMsk)
7	RW	0x0	GOUTNakEffMsk Global OUT NAK Effective Mask
6	RW	0x0	GINNakEffMsk Global Non-periodic IN NAK Effective Mask
5	RW	0x0	NPTxFEmpMsk Non-periodic TxFIFO Empty Mask
4	RW	0x0	RxFLvIMsk Receive FIFO Non-Empty Mask
3	RW	0x0	SofMsk Start of (micro)Frame Mask

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x0	OTGIntMsk OTG Interrupt Mask
1	RW	0x0	ModeMisMsk Mode Mismatch Interrupt Mask
0	RO	0x0	reserved

**USBOTG\_GRXSTSR**

Address: Operational Base + offset (0x001c)

Receive Status Debug Read Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:21	RO	0x0	FN Frame Number (Device Only) This is the least significant 4 bits of the (micro) frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.
20:17	RO	0x0	PktSts Packet Status Indicates the status of the received packet(Host Only) 4'b0010: IN data packet received 4'b0011: IN transfer completed (triggers an interrupt) 4'b0101: Data toggle error (triggers an interrupt) 4'b0111: Channel halted (triggers an interrupt) Others: Reserved Indicates the status of the received packet(Device only) 4'b0001: Global OUT NAK (triggers an interrupt) 4'b0010: OUT data packet received 4'b0011: OUT transfer completed (triggers an interrupt) 4'b0100: SETUP transaction completed (triggers an interrupt) 4'b0110: SETUP data packet received Others: Reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16:15	RO	0x0	DPID Data PID Indicates the Data PID of the received packet 2'b00: DATA0 2'b10: DATA1 2'b01: DATA2 2'b11: MDATA
14:4	RW	0x000	BCnt Byte Count Indicates the byte count of the received data packet.
3:0	RO	0x0	ChNum_EPNum Channel Number(Host) Endpoint Number(Device) (Host Only) Indicates the channel number to which the current received packet belongs. (Device Only) Indicates the endpoint number to which the current received packet belongs.

**USBOTG\_GRXSTSP**

Address: Operational Base + offset (0x0020)

Receive Status Read and Pop Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:21	RO	0x0	FN Frame Number (Device Only) This is the least significant 4 bits of the (micro) frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
20:17	RO	0x0	PktSts Packet Status Indicates the status of the received packet(Host Only) 4'b0010: IN data packet received 4'b0011: IN transfer completed (triggers an interrupt) 4'b0101: Data toggle error (triggers an interrupt) 4'b0111: Channel halted (triggers an interrupt) Others: Reserved Indicates the status of the received packet(Device only) 4'b0001: Global OUT NAK (triggers an interrupt) 4'b0010: OUT data packet received 4'b0011: OUT transfer completed (triggers an interrupt) 4'b0100: SETUP transaction completed (triggers an interrupt) 4'b0110: SETUP data packet received Others: Reserved
16:15	RO	0x0	DPID Data PID Indicates the Data PID of the received OUT data packet 2'b00: DATA0 2'b10: DATA1 2'b01: DATA2 2'b11: MDATA
14:4	RO	0x000	BCnt Byte Count Indicates the byte count of the received data packet.
3:0	RO	0x0	ChNum_EPNum Channel Number(Host) Endpoint Number(Device) (Host Only) Indicates the channel number to which the current received packet belongs. (Device Only) Indicates the endpoint number to which the current received packet belongs.

**USBOTG\_GRXFSIZ**

Address: Operational Base + offset (0x0024)

Receive FIFO Size Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	RxFDep Rx FIFO Depth This value is in terms of 32-bit words. Minimum value is 16, Maximum value is 32,768. The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth. If Enable Dynamic FIFO Sizing? was deselected, these flops are optimized, and reads return the power-on value. If Enable Dynamic FIFO Sizing? was selected , you can write a new value in this field, You can write a new value in this field. Programmed values must not exceed the power-on value.

**USBOTG\_GNPTXFSIZ**

Address: Operational Base + offset (0x0028)

Non-Periodic Transmit FIFO Size Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	NPTxFDep Non-periodic Tx FIFO For host mode, this field is always valid. For Device mode, this field is valid only when OTG_ENDED_TX_FIFO==0. This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 This field is determined by Enable Dynamic FIFO Sizing. OTG_DFIFO_DYNAMIC = 0: These flops are optimized, and reads return the Power on value. OTG_DFIFO_DYNAMIC = 1: The application can write a new value in this field. Programmed values must not exceed the power-on value. The power-on reset value of this field is specified by OTG_ENDED_TX_FIFO: OTG_ENDED_TX_FIFO = 0: The reset value is the Largest Non-periodic Tx Data FIFO Depth parameter, OTG_TX_NPERIO_DFIFO_DEPTH. OTG_ENDED_TX_FIFO = 1: The reset value is parameter OTG_TX_HNPERIO_DFIFO_DEPTH.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	<p>NPTxFStAddr Non-periodic Transmit RAM For host mode, this field is always valid. This field contains the memory start address for Non-periodic Transmit FIFO RAM. This field is determined by Enable Dynamic FIFO Sizing?(OTG_DFIFO_DYNAMIC):</p> <ul style="list-style-type: none"> <li>OTG_DFIFO_DYNAMIC = 0: These flops are optimized, and reads return the power-on value.</li> <li>OTG_DFIFO_DYNAMIC = 1: The application can write a new value in this field. Programmed values must not exceed the power-on value.</li> </ul> <p>The power-on reset value of this field is specified by Largest Rx Data FIFO Depth (parameter OTG_RX_DFIFO_DEPTH).</p>

**USBOTG\_GNPTXSTS**

Address: Operational Base + offset (0x002c)

Non-Periodic Transmit FIFO/Queue Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved
30:24	RO	0x00	<p>NPTxQTop Top of the Non-periodic Transmit Request Queue Entry in the Non-periodic Tx Request Queue that is currently being processed by the MAC.</p> <p>Bits [30:27]: Channel/endpoint number Bits [26:25]:</p> <ul style="list-style-type: none"> <li>2'b00: IN/OUT token</li> <li>2'b01: Zero-length transmit packet (device IN/host OUT)</li> <li>2'b10: PING/CSPLIT token</li> <li>2'b11: Channel halt command</li> </ul> <p>Bit [24]: Terminate (last entry for selected channel/endpoint)</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23:16	RO	0x00	<p>NPTxQSpAvail Non-periodic Transmit Request Queue Space Available Indicates the amount of free space available in the Non-periodic Transmit Request Queue. This queue holds both IN and OUT requests in Host mode. Device mode has only IN requests.</p> <p>8'h0: Non-periodic Transmit Request Queue is full 8'h1: 1 location available 8'h2: 2 locations available n: n locations available (0 &lt;=n &lt;= 8) Others: Reserved</p>
15:0	RO	0x0000	<p>NPTxFSpAvail Non-periodic TxFIFO Space Avail Indicates the amount of free space available in the Non-periodic TxFIFO. Values are in terms of 32-bit words.</p> <p>16'h0: Non-periodic TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 &lt;=n &lt;=32,768) 16'h8000: 32,768 words available Others: Reserved</p>

**USBOTG\_GI2CCTL**

Address: Operational Base + offset (0x0030)

I2C Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RWSC	0x0	<p>BsyDne I2C Busy/Done The application sets this bit to 1'b1 to start a request on the I2C interface. When the transfer is complete, the core de-asserts this bit to 1'b0. As long as the bit is set, indicating that the I2C interface is busy, the application cannot start another request on the interface.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
30	RW	0x0	RW Read/Write Indicator Indicates whether a read or write register transfer must be performed on the interface. Read/write bursting is not supported for registers. 1'b1: Read 1'b0: Write
29	RO	0x0	reserved
28	RW	0x1	I2CDatSe0 I2C DatSe0 USB Mode Selects the FS interface USB mode. 1'b1: VP_VM USB mode 1'b0: DAT_SE0 USB mode
27:26	RW	0x0	I2CDevAdr I2C Device Address Selects the address of the I2C Slave on the USB 1.1 full-speed serial transceiver that the core uses for OTG signaling. 2'b00: 7'h2C 2'b01: 7'h2D 2'b10: 7'h2E 2'b11: 7'h2F
25	RW	0x0	I2CSuspCtl I2C Suspend Control Selects how Suspend is connected to a full-speed transceiver in I2C mode. 1'b0: Use the dedicated utmi_suspend_n pin 1'b1: Use an I2C write to program the Suspend bit in the PHY register
24	RO	0x1	Ack I2C ACK Indicates whether an ACK response was received from the I2C Slave. This bit is valid when BsyDne is cleared by the core, after application has initiated an I2C access. 1'b0: NAK 1'b1: ACK
23	RW	0x0	I2CEn I2C Enable Enables the I2C Master to initiate I2C transactions on the I2C interface

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
22:16	RW	0x00	Addr I2C Address This is the 7-bit I2C device address used by software to access any external I2C Slave, including the I2C Slave on a USB 1.1 OTG full-speed serial transceiver. Software can change this address to access different I2C Slaves.
15:8	RW	0x00	RegAddr I2C Register Addr This field programs the address of the register to be read from or written to.
7:0	RW	0x00	RWData I2C Read/Write Data After a register read operation, this field holds the read data for the application. During a write operation, the application can use this register to program the write data to be written to a register. During writes, this field holds the write data.

**USBOTG\_GPVNDCTL**

Address: Operational Base + offset (0x0034)

PHY Vendor Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RWSC	0x0	DisUlpiDrvr Disable ULPI Drivers This field is used only if the Carkit interface was enabled (OTG_ULPI_CARKIT = 1). Otherwise, reads return 0. The application sets this bit when it has finished processing the ULPI Carkit Interrupt (GINTSTS.ULPICKINT). When set, the Otg core disables drivers for output signals and masks input signal for the ULPI interface. Otg clears this bit before enabling the ULPI interface.
30:28	RO	0x0	reserved
27	RWSC	0x0	VStsDone VStatus Done The core sets this bit when the vendor control access is done. This bit is cleared by the core when the application sets the New Register Request bit (bit 25).

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
26	RO	0x0	VStsBsy VStatus Busy The core sets this bit when the vendor control access is in progress and clears this bit when done.
25	R/WSC	0x0	NewRegReq New Register Request The application sets this bit for a new vendor control access.
24:23	RO	0x0	reserved
22	RW	0x0	RegWr Register Write Set this bit for register writes, and clear it for register reads.
21:16	RW	0x00	RegAddr Register Address The 6-bit PHY register address for immediate PHY Register Set access. Set to 6'h2F for Extended PHY Register Set access.
15:8	RW	0x00	VCtrl UTMI+ Vendor Control Register Address The 4-bit register address a vendor defined 4-bit parallel output bus. Bits 11:8 of this field are placed on utmi_vcontrol[3:0]. ULPI Extended Register Address (ExtRegAddr) The 6-bit PHY extended register address.
7:0	RW	0x00	RegData Register Data Contains the write data for register write. Read data for register read, valid when VStatus Done is set.

**USBOTG\_GGPIO**

Address: Operational Base + offset (0x0038)

General Purpose Input/Output Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	GPO General Purpose Output This field is driven as an output from the core, gp_o[15:0]. The application can program this field to determine the corresponding value on the gp_o[15:0] output.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RO	0x0000	GPI General Purpose Input This field's read value reflects the gp_i[15:0] core input value.

**USBOTG\_GUID**

Address: Operational Base + offset (0x003c)

User ID Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	UserID Application-programmable ID field.

**USBOTG\_GSNPSID**

Address: Operational Base + offset (0x0040)

Core ID Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00004f54	CoreID Release number of the core being used

**USBOTG\_GHWCFG1**

Address: Operational Base + offset (0x0044)

User HW Config1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	epdir Endpoint Direction This 32-bit field uses two bits per endpoint to determine the endpoint direction. Endpoint Bits [31:30]: Endpoint 15 direction Bits [29:28]: Endpoint 14 direction ... Bits [3:2]: Endpoint 1 direction Bits[1:0]: Endpoint 0 direction (always BIDIR) Direction 2'b00: BIDIR (IN and OUT) endpoint 2'b01: IN endpoint 2'b10: OUT endpoint 2'b11: Reserved

**USBOTG\_GHWCFG2**

Address: Operational Base + offset (0x0048)

User HW Config2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	OTG_ENABLE_IC_USB IC_USB mode specified for mode of operation (parameter OTG_ENABLE_IC_USB). To choose IC_USB_MODE, both OTG_FSPHY_INTERFACE and OTG_ENABLE_IC_USB must be 1.
30:26	RO	0x00	TknQDepth Device Mode IN Token Sequence Learning Queue Depth Range: 0-30
25:24	RO	0x0	PTxQDepth Host Mode Periodic Request Queue Depth 2'b00: 2 2'b01: 4 2'b10: 8 Others: Reserved
23:22	RO	0x0	NPTxQDepth Non-periodic Request Queue Depth 2'b00: 2 2'b01: 4 2'b10: 8 Others: Reserved
21	RO	0x0	reserved
20	RO	0x0	MultiProcIntrpt Multi-Processor Interrupt Enabled 1'b0: No 1'b1: Yes
19	RO	0x0	DynFifoSizing Dynamic FIFO Sizing Enabled 1'b0: No 1'b1: Yes
18	RO	0x0	PerioSupport Periodic OUT Channels Supported in Host Mode 1'b0: No 1'b1: Yes
17:14	RO	0x0	NumHstChnl Number of Host Channels Indicates the number of host channels supported by the core in Host mode. The range of this field is 0-15: 0 specifies 1 channel, 15 specifies 16 channels.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13:10	RO	0x0	NumDevEps Number of Device Endpoints Indicates the number of device endpoints supported by the core in Device mode in addition to control endpoint 0. The range of this field is 1-15.
9:8	RO	0x0	FSPhyType Full-Speed PHY Interface Type 2'b00: Full-speed interface not supported 2'b01: Dedicated full-speed interface 2'b10: FS pins shared with UTMI+ pins 2'b11: FS pins shared with ULPI pins
7:6	RO	0x0	HSPhyType High-Speed PHY Interface Type 2'b00: High-Speed interface not supported 2'b01: UTMI+ 2'b10: ULPI 2'b11: UTMI+ and ULPI
5	RO	0x0	SingPnt Point-to-Point 1'b0: Multi-point application (hub and split support) 1'b1: Single-point application (no hub and no split support)
4:3	RO	0x0	OtgArch Architecture 2'b00: Slave-Only 2'b01: External DMA 2'b10: Internal DMA Others: Reserved
2:0	RO	0x0	OtgMode Mode of Operation 3'b000: HNP- and SRP-Capable OTG (Host and Device) 3'b001: SRP-Capable OTG (Host and Device) 3'b010: Non-HNP and Non-SRP Capable OTG (Host and Device) 3'b011: SRP-Capable Device 3'b100: Non-OTG Device 3'b101: SRP-Capable Host 3'b110: Non-OTG Host Others: Reserved

**USBOTG\_GHWCFG3**

Address: Operational Base + offset (0x004c)

## User HW Config3 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	DfifoDepth DFIFO Depth This value is in terms of 32-bit words. Minimum value is 32 Maximum value is 32,768
15	RO	0x0	OTG_ENABLE_LPM LPM mode specified for Mode of Operation (parameter OTG_ENABLE_LPM).
14	RO	0x0	OTG_BC_SUPPORT This bit indicates the HS OTG controller support for Battery Charger. 1'b0: No Battery Charger Support 1'b1: Battery Charger support present.
13	RO	0x0	OTG_ENABLE_HSIC HSIC mode specified for Mode of Operation (parameter OTG_ENABLE_HSIC). Value Range: 0-1 1'b1: HSIC-capable with shared UTMI PHY interface 1'b0: Non-HSIC-capable
12	RO	0x0	OTG_AD_P_SUPPORT This bit indicates whether ADP logic is present within or external to the HS OTG controller 1'b0: No ADP logic present with HSOTG controller 1'b1: ADP logic is present along with HSOTG controller.
11	RO	0x0	RstType Reset Style for Clocked always Blocks in RTL 1'b0: Asynchronous reset is used in the core 1'b1: Synchronous reset is used in the core
10	RO	0x0	OptFeature Optional Features Removed Indicates whether the User ID register, GPIO interface ports, and SOF toggle and counter ports were removed for gate count optimization by enabling Remove Optional Features? 1'b0: No 1'b1: Yes

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RO	0x0	VndctlSupt Vendor Control Interface Support 1'b0: Vendor Control Interface is not available on the core. 1'b1: Vendor Control Interface is available.
8	RO	0x0	I2CIntSel I2C Selection 1'b0: I2C Interface is not available on the core. 1'b1: I2C Interface is available on the core.
7	RO	0x0	OtgEn OTG Function Enabled The application uses this bit to indicate the Otg core's OTG capabilities. 1'b0: Not OTG capable 1'b1: OTG Capable
6:4	RO	0x0	PktSizeWidth Width of Packet Size Counters 3'b000: 4 bits 3'b001: 5 bits 3'b010: 6 bits 3'b011: 7 bits 3'b100: 8 bits 3'b101: 9 bits 3'b110: 10 bits Others: Reserved
3:0	RO	0x0	XferSizeWidth Width of Transfer Size Counters 4'b0000: 11 bits 4'b0001: 12 bits ... 4'b1000: 19 bits Others: Reserved

**USBOTG\_GHWCFG4**

Address: Operational Base + offset (0x0050)

User HW Config4 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	SGDMA Scatter/Gather DMA 1'b1: Dynamic configuration
30	RO	0x0	SGDMACon Scatter/Gather DMA configuration 1'b0: Non-Scatter/Gather DMA configuration 1'b1: Scatter/Gather DMA configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
29:26	RO	0x0	INEps Number of Device Mode IN Endpoints Including Control Endpoint Range 0 -15 0:1 IN Endpoint 1:2 IN Endpoints .... 15:16 IN Endpoints
25	RW	0x0	DedFifoMode Enable Dedicated Transmit FIFO for device IN Endpoints 1'b0: Dedicated Transmit FIFO Operation not enabled. 1'b1: Dedicated Transmit FIFO Operation enabled.
24	RW	0x0	SessEndFltr session_end Filter Enabled 1'b0: No filter 1'b1: Filter
23	RW	0x0	BValidFltr "b_valid" Filter Enabled 1'b0: No filter 1'b1: Filter
22	RO	0x0	AValidFltr "a_valid" Filter Enabled 1'b0: No filter 1'b1: Filter
21	RO	0x0	VBusValidFltr "vbus_valid" Filter Enabled 1'b0: No filter 1'b1: Filter
20	RO	0x0	IddgFltr "iddig" Filter Enable 1'b0: No filter 1'b1: Filter
19:16	RO	0x0	NumCtlEps Number of Device Mode Control Endpoints in Addition to Endpoint Range: 0-15

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:14	RO	0x0	PhyDataWidth UTMI+ PHY/ULPI-to-Internal UTMI+ Wrapper Data Width When a ULPI PHY is used, an internal wrapper converts ULPI to UTMI+. 2'b00: 8 bits 2'b01: 16 bits 2'b10: 8/16 bits, software selectable Others: Reserved
13:7	RO	0x0	reserved
6	RO	0x0	EnHiber Enable Hibernation 1'b0: Hibernation feature not enabled 1'b1: Hibernation feature enabled
5	RO	0x0	AhbFreq Minimum AHB Frequency Less Than 60 MHz 1'b0: No 1'b1: Yes
4	RO	0x0	EnParPwrDown Enable Partial Power Down 1'b0: Partial Power Down Not Enabled 1'b1: Partial Power Down Enabled
3:0	RO	0x0	NumDevPerioEps Number of Device Mode Periodic IN Endpoints Range: 0-15

**USBOTG\_GLPMCFG**

Address: Operational Base + offset (0x0054)

Core LPM Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	<p>InvSelHsic HSIC-Invert Select HSIC</p> <p>The application uses this bit to control the Otg core HSIC enable/disable. This bit overrides and functionally inverts the if_sel_hsic input port signal. If the core operates as non-HSIC-capable, it can only connect to non-HSIC-capable PHYs. If the core operates as HSIC-capable, it can only connect to HSIC capable PHYs. If the if_sel_hsic input signal is1:</p> <ul style="list-style-type: none"> <li>1'b1: HSIC capability is not enabled.</li> <li>1'b0: HSIC capability is enabled,</li> </ul> <p>If InvSelHsic = 1'b0: HSIC capability is enabled. If the if_sel_hsic input signal is 0:</p> <ul style="list-style-type: none"> <li>1'b1: HSIC capability is enabled,</li> <li>1'b0: HSIC capability is not enabled.</li> </ul> <p>This bit is writable only if an HSIC mode was specified for Mode of Operation (parameter OTG_ENABLE_HSIC). This bit is valid only if OTG_ENABLE_HSIC is enabled.</p>
30	RW	0x0	<p>HSICCon HSIC-Connect</p> <p>The application must use this bit to initiate the HSIC Attach sequence. Host Mode: Once this bit is set, the host core configures to drive the HSIC Idle state (STROBE = 1 &amp; DATA = 0) on the bus. It then waits for the device to initiate the Connect sequence. Device Mode: Once this bit is set, the device core waits for the HSIC Idle line state on the bus. Upon receiving the Idle line state, it initiates the HSIC Connect sequence. This bit is valid only if OTG_ENABLE_HSIC is 1, if_sel_hsic = 1 and InvSelHSIC is 0. Otherwise, it is read-only.</p>
29:28	RO	0x0	reserved
27:25	RO	0x0	<p>LPM_RetryCnt_Sts LPM Retry Count Status</p> <p>Number of LPM host retries remaining to be transmitted for the current LPM sequence.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
24	RW	0x0	SndLPM Send LPM Transaction Host Mode: When the application software sets this bit, an LPM transaction containing two tokens, EXT and LPM, is sent. The hardware clears this bit once a valid response (STALL, NYET, or ACK) is received from the device or the core has finished transmitting the programmed number of LPM retries. Note: This bit must only be set when the host is connected to a local port.
23:21	RWSC	0x0	LPM_Retry_Cnt LPM Retry Count When the device gives an ERROR response, this is the number of additional LPM retries that the host performs until a valid device response (STALL, NYET, or ACK) is received.
20:17	RW	0x0	LPM_Chnl_Indx LPM Channel Index The channel number on which the LPM transaction must be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and endpoint number programmed in the corresponding channel into the LPM transaction.
16	RO	0x0	L1ResumeOK Sleep State Resume OK Indicates that the application or host can start a resume from the Sleep state. This bit is valid in the LPM Sleep (L1) state. It is set in Sleep mode after a delay of 50 us (TL1Residency). The bit is reset when SlpSts is 0 1'b1: The application/core can start resume from the Sleep state 1'b0: The application/core cannot start resume from the Sleep state

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RO	0x0	<p>SlpSts Port Sleep Status</p> <p>Device Mode: This bit is set as long as a Sleep condition is present on the USB bus. The core enters the Sleep state when an ACK response is sent to an LPM transaction and the timer TL1TokenRetry. has expired. To stop the PHY clock, the application must set the Port Clock Stop bit, which asserts the PHY Suspend input pin. The application must rely on SlpSts and not ACK in CoreL1Res to confirm transition into sleep.</p> <p>The core comes out of sleep: When there is any activity on the USB line_state</p> <p>When the application writes to the Remote Wakeup Signaling bit in the Device Control register (DCTL.RmtWkUpSig) or when the application resets or soft-disconnects the device.</p> <p>Host Mode: The host transitions to the Sleep (L1) state as a sideeffect of a successful LPM transaction by the core to the local port with an ACK response from the device. The read value of this bit reflects the port's current sleep status. The core clears this bit after: The core detects a remote L1 Wakeup signal The application sets the Port Reset bit or the Port L1Resume bit in the HPRT register or The application sets the L1Resume/ Remote Wakeup Detected Interrupt bit or Disconnect Detected Interrupt bit in the Core Interrupt register (GINTSTS.L1WkUpInt or GINTSTS.DisconnInt, respectively).</p> <p>Values:</p> <ul style="list-style-type: none"> <li>1'b0: Core not in L1</li> <li>1'b1: Core in L1</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>																																																			
14:13	RO	0x0	<p>CoreL1Res LPM Response Device Mode: The core's response to the received LPM transaction is reflected in these two bits. Host Mode: The handshake response received from the local device for LPM transaction.</p> <p>2'b11: ACK 2'b10: NYET 2'b01: STALL 2'b00: ERROR (No handshake response)</p>																																																			
12:8	RW	0x00	<p>HIRD_Thres HIRD Threshold Device Mode: The core asserts L1SuspendM to put the PHY into Deep Low-Power mode in L1 when the HIRD value is greater than or equal to the value defined in this field (GLPMCFG.HIRD_Thres[3:0]), and HIRD_Thres[4] is set to 1'b1. Host Mode: The core asserts L1SuspendM to put the PHY into Deep Low-Power mode in L1 when HIRD_Thres[4] is set to 1'b1. HIRD_Thres[3:0] specifies the time for which resume signaling is to be reflected by the host TL1HubDrvResume2) on the USB when it detects device-initiated resume. HIRD_Thres must not be programmed with a value greater than 4'b1100 in Host mode, because this exceeds maximum <math>T_{L1HubDrvResume2}</math>.</p> <table> <thead> <tr> <th>No</th> <th>HIRD_Thres[3:0]</th> <th>resume time(us)</th> </tr> </thead> <tbody> <tr><td>1</td><td>4'b0000</td><td>60</td></tr> <tr><td>2</td><td>4'b0001</td><td>135</td></tr> <tr><td>3</td><td>4'b0010</td><td>210</td></tr> <tr><td>4</td><td>4'b0011</td><td>285</td></tr> <tr><td>5</td><td>4'b0100</td><td>360</td></tr> <tr><td>6</td><td>4'b0101</td><td>435</td></tr> <tr><td>7</td><td>4'b0110</td><td>510</td></tr> <tr><td>8</td><td>4'b0111</td><td>585</td></tr> <tr><td>9</td><td>4'b1000</td><td>660</td></tr> <tr><td>10</td><td>4'b1001</td><td>735</td></tr> <tr><td>11</td><td>4'b1010</td><td>810</td></tr> <tr><td>12</td><td>4'b1011</td><td>885</td></tr> <tr><td>13</td><td>4'b1100</td><td>960</td></tr> <tr><td>14</td><td>4'b1101</td><td>invalid</td></tr> <tr><td>15</td><td>4'b1110</td><td>invalid</td></tr> <tr><td>16</td><td>4'b1111</td><td>invalid</td></tr> </tbody> </table>	No	HIRD_Thres[3:0]	resume time(us)	1	4'b0000	60	2	4'b0001	135	3	4'b0010	210	4	4'b0011	285	5	4'b0100	360	6	4'b0101	435	7	4'b0110	510	8	4'b0111	585	9	4'b1000	660	10	4'b1001	735	11	4'b1010	810	12	4'b1011	885	13	4'b1100	960	14	4'b1101	invalid	15	4'b1110	invalid	16	4'b1111	invalid
No	HIRD_Thres[3:0]	resume time(us)																																																				
1	4'b0000	60																																																				
2	4'b0001	135																																																				
3	4'b0010	210																																																				
4	4'b0011	285																																																				
5	4'b0100	360																																																				
6	4'b0101	435																																																				
7	4'b0110	510																																																				
8	4'b0111	585																																																				
9	4'b1000	660																																																				
10	4'b1001	735																																																				
11	4'b1010	810																																																				
12	4'b1011	885																																																				
13	4'b1100	960																																																				
14	4'b1101	invalid																																																				
15	4'b1110	invalid																																																				
16	4'b1111	invalid																																																				

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	<p>EnbISlpM Enable utmi_sleep_n For ULPI interface: The application uses this bit to write to the function control [7] in the L1 state, to enable the PHY to go into Low Power mode. For the host, this bit is valid only in Local Device mode.</p> <p>1'b0: Writes to the ULPI Function Control Bit [7] are disabled.</p> <p>1'b1: The core is enabled to write to the ULPI Function Control Bit [7], which enables the PHY to enter Low-Power mode.</p> <p>Note: When a ULPI interface is configured, enabling this bit results in a write to Bit 7 of the ULPI Function Control register. The ULPI PHY supports writing to this bit, and in the L1 state asserts SleepM when utmi_l1_suspend_n cannot be asserted.</p> <p>When a ULPI interface is configured, this bit must always be set if you are using the ULPI PHY. Note: For ULPI interface, do not clear this bit during the resume. For all other interfaces: The application uses this bit to control utmi_sleep_n assertion to the PHY in the L1 state. For the host, this bit is valid only in Local Device mode.</p> <p>1'b0: utmi_sleep_n assertion from the core is not transferred to the external PHY.</p> <p>1'b1: utmi_sleep_n assertion from the core is transferred to the external PHY when utmi_l1_suspend_n cannot be asserted.</p>
6	RW	0x0	<p>bRemoteWake RemoteWakeEnable Host Mode: The remote wakeup value to be sent in the LPM transaction's wIndex field. Device Mode: This field is updated with the received bRemoteWake LPM token's bmAttribute when an ACK/NYET/STALL response is sent to an LPM transaction.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>																																																			
5:2	RW	0x0	<p>HIRD Host-Initiated Resume Duration Host Mode: The value of HIRD to be sent in an LPM transaction. This value is also used to initiate resume for a duration <math>T_{L1HubDrvResume1}</math> for host initiated resume. Device Mode: This field is updated with the Received LPM Token HIRD bmAttribute when an ACK/NYET/STALL response is sent to an LPM transaction</p> <table> <thead> <tr> <th>SI. No</th> <th>HIRD[3:0]</th> <th>THIRD (us)</th> </tr> </thead> <tbody> <tr><td>1</td><td>4'b0000</td><td>50</td></tr> <tr><td>2</td><td>4'b0001</td><td>125</td></tr> <tr><td>3</td><td>4'b0010</td><td>200</td></tr> <tr><td>4</td><td>4'b0011</td><td>275</td></tr> <tr><td>5</td><td>4'b0100</td><td>350</td></tr> <tr><td>6</td><td>4'b0101</td><td>425</td></tr> <tr><td>7</td><td>4'b0110</td><td>500</td></tr> <tr><td>8</td><td>4'b0111</td><td>575</td></tr> <tr><td>9</td><td>4'b1000</td><td>650</td></tr> <tr><td>10</td><td>4'b1001</td><td>725</td></tr> <tr><td>11</td><td>4'b1010</td><td>800</td></tr> <tr><td>12</td><td>4'b1011</td><td>875</td></tr> <tr><td>13</td><td>4'b1100</td><td>950</td></tr> <tr><td>14</td><td>4'b1101</td><td>1025</td></tr> <tr><td>15</td><td>4'b1110</td><td>1100</td></tr> <tr><td>16</td><td>4'b1111</td><td>1175</td></tr> </tbody> </table>	SI. No	HIRD[3:0]	THIRD (us)	1	4'b0000	50	2	4'b0001	125	3	4'b0010	200	4	4'b0011	275	5	4'b0100	350	6	4'b0101	425	7	4'b0110	500	8	4'b0111	575	9	4'b1000	650	10	4'b1001	725	11	4'b1010	800	12	4'b1011	875	13	4'b1100	950	14	4'b1101	1025	15	4'b1110	1100	16	4'b1111	1175
SI. No	HIRD[3:0]	THIRD (us)																																																				
1	4'b0000	50																																																				
2	4'b0001	125																																																				
3	4'b0010	200																																																				
4	4'b0011	275																																																				
5	4'b0100	350																																																				
6	4'b0101	425																																																				
7	4'b0110	500																																																				
8	4'b0111	575																																																				
9	4'b1000	650																																																				
10	4'b1001	725																																																				
11	4'b1010	800																																																				
12	4'b1011	875																																																				
13	4'b1100	950																																																				
14	4'b1101	1025																																																				
15	4'b1110	1100																																																				
16	4'b1111	1175																																																				

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	<p>AppL1Res LPM response programmed by application Handshake response to LPM token pre-programmed by device application software. The response depends on GLPMCFG.LPMCap. If GLPMCFG.LPMCap is 1'b0, the core always responds with a NYET. If GLPMCFG.LPMCap is 1'b1, the core responds as follows:</p> <p>1'b1: ACK. Even though an ACK is pre-programmed, the core responds with an ACK only on a successful LPM transaction. The LPM transaction is successful if: There are no PID/CRC5 errors in both the EXT token and the LPM token (else ERROR); A valid bLinkState = 0001B (L1) is received in the LPM transaction (else STALL); No data is pending in the Transmit queue (else NYET)</p> <p>1'b0: NYET. The pre-programmed software bit is overridden for response to LPM token when: (1) The received bLinkState is not L1 (STALL response); (2) An error is detected in either of the LPM token packets due to corruption (ERROR response).</p>
0	RW	0x0	<p>LPMCap LPM-Capable The application uses this bit to control the OTG core LPM capabilities. If the core operates as a non-LPM-capable host, it cannot request the connected device/hub to activate LPM mode. If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions.</p> <p>1'b0: LPM capability is not enabled. 1'b1: LPM capability is enabled. This bit is writable only if an LPM mode was specified for Mode of Operation (parameter OTG_ENABLE_LPM). Otherwise, reads return 0.</p>

**USBOTG\_GPWDN**

Address: Operational Base + offset (0x0058)

Global Power Down Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
28:24	RO	0x00	MultValIdBC Multi Valued ID pin Battery Charger ACA inputs in the following order: Bit 26 - rid_float. Bit 25 - rid_gnd Bit 24 - rid_a Bit 23 - rid_b Bit 22 - rid_c These bits are present only if OTG_BC_SUPPORT = 1. Otherwise, these bits are reserved and will read 5'h0.
23	W1C	0x0	ADPInt ADP Interrupt This bit is set whenever there is a ADP event.
22	RO	0x0	BSessVld B Session Valid This field reflects the B session valid status signal from the PHY. 1'b0: B-Valid is 0. 1'b1: B-Valid is 1. This bit is valid only when GPWRDN.PMUActv is 1.
21	RO	0x0	IDDIG This bit indicates the status of the signal IDDIG. The application must read this bit after receiving GPWRDN.StsChngInt and decode based on the previous value stored by the application. Indicates the current mode. 1'b1: Device mode 1'b0: Host mode This bit is valid only when GPWRDN.PMUActv is 1.
20:19	RO	0x0	LineState This field indicates the current linestate on USB as seen by the PMU module. 2'b00: DM = 0, DP = 0. 2'b01: DM = 0, DP = 1. 2'b10: DM = 1, DP = 0. 2'b11: Not-defined. This bit is valid only when GPWRDN.PMUActv is 1.
18	RW	0x0	StsChngIntMsk Mask For StsChng Interrupt

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17	W1C	0x0	<p>StsChngInt This field indicates a status change in either the IDDIG or BSessVld signal.</p> <p>1'b0: No Status change 1'b1: status change detected</p> <p>After receiving this interrupt the application should read the GPWRDN register and interpret the change in IDDIG or BSesVld with respect to the previous value stored by the application.</p>
16	RW	0x0	<p>SRPDetectMsk Mask For SRPDetect Interrupt</p>
15	W1C	0x0	<p>SRPDetect This field indicates that SRP has been detected by the PMU. This field generates an interrupt. After detecting SRP during hibernation the application should not restore the core. The application should get into the initialization process.</p> <p>1'b0: SRP not detected 1'b1: SRP detected</p>
14	RW	0x0	<p>ConnDetMsk Mask for ConnectDet interrupt</p> <p>This bit is valid only when OTG_EN_PWRROPT = 2.</p>
13	W1C	0x0	<p>ConnectDet This field indicates that a new connect has been detected</p> <p>1'b0: Connect not detected 1'b1: Connect detected</p> <p>This bit is valid only when OTG_EN_PWRROPT = 2.</p>
12	RW	0x0	<p>DisconnectDetectMsk Mask For DisconnectDetect Interrupt</p> <p>This bit is valid only when OTG_EN_PWRROPT = 2.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	W1C	0x0	<p>DisconnectDetect</p> <p>This field indicates that Disconnect has been detected by the PMU. This field generates an interrupt. After detecting disconnect during hibernation the application must not restore the core, but instead start the initialization process.</p> <p>1'b0: Disconnect not detected 1'b1: Disconnect detected</p> <p>This bit is valid only when OTG_EN_PWR0PT = 2.</p>
10	RW	0x0	<p>ResetDetMsk</p> <p>Mask For ResetDetected interrupt. This bit is valid only when OTG_EN_PWR0PT = 2.</p>
9	W1C	0x0	<p>ResetDetected</p> <p>This field indicates that Reset has been detected by the PMU module. This field generates an interrupt.</p> <p>1'b0: Reset Not Detected 1'b1: Reset Detected</p> <p>This bit is valid only when OTG_EN_PWR0PT = 2.</p>
8	RW	0x0	<p>LineStageChangeMsk</p> <p>Mask For LineStateChange interrupt</p> <p>This bit is valid only when OTG_EN_PWR0PT = 2.</p>
7	W1C	0x0	<p>LnStsChng</p> <p>Line State Change</p> <p>This interrupt is asserted when there is a Linestate Change detected by the PMU. The application should read GPWRDN.Linestate to determine the current linestate on USB.</p> <p>1'b0: No LineState change on USB 1'b1: LineState change on USB</p> <p>This bit is valid only when GPWRDN.PMUActv is 1. This bit is valid only when OTG_EN_PWR0PT = 2.</p>
6	RW	0x0	<p>DisableVBUS</p> <p>The application should program this bit if HPRT0.PrtPwr was programmed to 0 before entering Hibernation. This is to indicate PMU whether session was ended before entering Hibernation.</p> <p>1'b0: HPRT0.PrtPwr was not programmed to 0. 1'b1: HPRT0.PrtPwr was programmed to 0.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	<p>PwrDnSwtch Power Down Switch This bit indicates to the OTG core VDD switch is in ON/OFF state 1'b0: OTG is in ON state 1'b1: OTG is in OFF state Note: This bit must not be written to during normal mode of operation.</p>
4	RW	0x0	<p>PwrDnRst_n Power Down ResetN The application must program this bit to reset the DWC OTG core during the Hibernation exit process or during ADP when powering up the core (in case the DWC OTG core was powered off during ADP process). 1'b1: OTG is in normal operation 1'b0: reset OTG Note: This bit must not be written to during normal mode of operation.</p>
3	RW	0x0	<p>PwrDnClmp Power Down Clamp The application must program this bit to enable or disable the clamps to all the outputs of the DWC OTG core module to prevent the corruption of other active logic. 1'b0: Disable PMU power clamp 1'b1: Enable PMU power clamp</p>
2	RW	0x0	<p>Restore The application should program this bit to enable or disable restore mode from the PMU module. 1'b0: OTG in normal mode of operation 1'b1: OTG in restore mode Note: This bit must not be written to during normal mode of operation. This bit is valid only when OTG_EN_PWRROPT = 2.</p>
1	RW	0x0	<p>PMUActv PMU Active This is bit is to enable or disable the PMU logic. 1'b0: Disable PMU module 1'b1: Enable PMU module Note: This bit must not be written to during normal mode of operation.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	<p>PMUIntSel PMU Interrupt Select When the hibernation functionality is selected using the configuration option OTG_EN_PWR_OPT = 2, a write to this bit with 1'b1 enables the PMU to generate interrupts to the application. During this state all interrupts from the core module are blocked to the application. Note: This bit must be set to 1'b1 before the core is put into hibernation</p> <p>1'b0: Internal OTG_core interrupt is selected 1'b1: the external OTG_pmu interrupt is selected</p> <p>Note: This bit must not be written to during normal mode of operation.</p>

**USBOTG\_GDFIFO CFG**

Address: Operational Base + offset (0x005c)

Global DFIFO Software Config Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>EPInfoBaseAddr This field provides the start address of the EP info controller.</p>
15:0	RW	0x0000	<p>GDFIFO Cfg This field is for dynamic programming of the DFIFO Size. This value takes effect only when the application programs a non-zero value to this register. The core does not have any corrective logic if the FIFO sizes are programmed incorrectly.</p>

**USBOTG\_GADPCTL**

Address: Operational Base + offset (0x0060)

ADP Timer, Control and Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:27	RWSC	0x0	<p>AR Access Request 2'b00 Read/Write Valid (updated by the core) 2'b01 Read 2'b10 Write 2'b11 Reserved</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
26	RW	0x0	AdpTmoutMsk ADP Timeout Interrupt Mask When this bit is set, it unmasks the interrupt because of AdpTmoutInt. This bit is valid only if OTG_Ver = 1'b1(GOTGCTL[20]).
25	RW	0x0	AdpSnsIntMsk ADP Sense Interrupt Mask When this bit is set, it unmasks the interrupt due to AdpSnsInt. This bit is valid only if OTG_Ver = 1'b1(GOTGCTL[20]).
24	RW	0x0	AdpPrbIntMsk ADP Probe Interrupt Mask When this bit is set, it unmasks the interrupt due to AdpPrbInt. This bit is valid only if OTG_Ver = 1'b1(GOTGCTL[20]).
23	W1C	0x0	AdpTmoutInt ADP Timeout Interrupt This bit is relevant only for an ADP probe. When this bit is set, it means that the ramp time has completed (GADPCTL.RTIM has reached its terminal value of 0x7FF). This is a debug feature that allows software to read the ramp time after each cycle. This bit is valid only if OTG_Ver = 1'b1.
22	W1C	0x0	AdpSnsInt ADP Sense Interrupt When this bit is set, it means that the VBUS voltage is greater than VadpSns value or VadpSns is reached. This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).
21	W1C	0x0	AdpPrbInt ADP Probe Interrupt When this bit is set, it means that the VBUS voltage is greater than VadpPrb or VadpPrb is reached. This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).
20	RW	0x0	ADPEN ADP Enable When set, the core performs either ADP probing or sensing based on EnaPrb or EnaSns. This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19	RWSC	0x0	ADPRes ADP Reset When set, ADP controller is reset. This bit is auto-cleared after the reset procedure is complete in ADP controller. This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).
18	RW	0x0	EnaSns Enable Sense When programmed to 1'b1, the core performs a sense operation. This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).
17	RW	0x0	EnaPrb Enable Probe When programmed to 1'b1, the core performs a probe operation. This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).
16:6	RO	0x000	RTIM RAMP TIME These bits capture the latest time it took for VBUS to ramp from VADP_SINK to VADP_PRB. The bits are defined in units of 32 kHz clock cycles as follows: 3'b000 - 1 cycles 3'b001 - 2 cycles 3'b010 - 3 cycles and so on till 0x7FF - 2048 cycles A time of 1024 cycles at 32 kHz corresponds to a time of 32 msec. (Note for scaledown ramp_timeout = prb_delta == 2'b00 => 200 cycles prb_delta == 2'b01 => 100 cycles prb_delta == 2'b01 => 50 cycles prb_delta == 2'b01 => 25 cycles.)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:4	RW	0x0	<p>PrbPer Probe Period</p> <p>These bits sets the TadpPrd as follows:</p> <ul style="list-style-type: none"> <li>2'b00 - 0.625 to 0.925 sec (typical 0.775 sec)</li> <li>2'b01 - 1.25 to 1.85 sec (typical 1.55 sec)</li> <li>2'b10 - 1.9 to 2.6 sec (typical 2.275 sec)</li> <li>2'b11 - Reserved</li> </ul> <p>(PRB_PER is also scaledown prb_per== 2'b00 =&gt; 400 ADP clocks prb_per== 2'b01 =&gt; 600 ADP clocks prb_per== 2'b10 =&gt; 800 ADP clocks prb_per==2'b11 =&gt; 1000 ADP clocks)</p>
3:2	RW	0x0	<p>PrbDelta Probe Delta</p> <p>These bits set the resolution for RTIM value. The bits are defined in units of 32 kHz clock cycles as follows:</p> <ul style="list-style-type: none"> <li>2'b00 - 1 cycles</li> <li>2'b01 - 2 cycles</li> <li>2'b10 - 3 cycles</li> <li>2'b11 - 4 cycles</li> </ul> <p>For example if this value is chosen to 2'b01, it means that RTIM increments for every three 32Khz clock cycles.</p>
1:0	RW	0x0	<p>PrbDschg Probe Discharge</p> <p>These bits set the times for TadpDschg. These bits are defined as follows:</p> <ul style="list-style-type: none"> <li>2'b00 4 msec (Scaledown 2 32Khz clock cycles)</li> <li>2'b01 8 msec (Scaledown 4 32Khz clock cycles)</li> <li>2'b10 16 msec (Scaledown 8 32Khz clock cycles)</li> <li>2'b11 32 msec (Scaledown 16 32Khz clock cycles)</li> </ul>

**USBOTG\_HPTXFSIZ**

Address: Operational Base + offset (0x0100)

Host Periodic Transmit FIFO Size Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>PTxFSize Host Periodic TxFIFO Depth This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest Host Mode Periodic Tx Data FIFO Depth (parameter OTG_TX_HPERIO_DFIFO_DEPTH). If Enable Dynamic FIFO Sizing? Was deselected (parameter OTG_DFIFO_DYNAMIC = 0), these flops are optimized, and reads return the power-on value. If Enable Dynamic FIFO Sizing? was selected (parameter OTG_DFIFO_DYNAMIC = 1), you can write a new value in this field. Programmed values must not exceed the power-on value set .</p>
15:0	RW	0x0000	<p>PTxFSAddr Host Periodic TxFIFO Start Address The power-on reset value of this register is the sum of the Largest Rx Data FIFO Depth and Largest Non-periodic Tx Data FIFO Depth specified. These parameters are: In shared FIFO operation: OTG_RX_DFIFO_DEPTH + OTG_TX_NPERIO_DFIFO_DEPTH. In dedicated FIFO mode: OTG_RX_DFIFO_DEPTH + OTG_TX_HNPERIO_DFIFO_DEPTH. If Enable Dynamic FIFO Sizing? was deselected (parameter OTG_DFIFO_DYNAMIC = 0 ), these flops are optimized, and reads return the power-on value. If Enable Dynamic FIFO Sizing? was selected (parameter OTG_DFIFO_DYNAMIC = 1), you can write a new value in this field. Programmed values must not exceed the power-on value.</p>

**USBOTG\_DIEPTXFn**

Address: Operational Base + offset (0x0104+0x4\*(n-1)), n = 1 - 15

Device Periodic Transmit FIFO-n Size Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	<p>INEP1TxFDep IN Endpoint TxFIFO Depth This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest IN Endpoint FIFO number Depth (parameter OTG_TX_DINEP_DFIFO_DEPTH_n)(0 &lt; n &lt;= 15). If Enable Dynamic FIFO Sizing? was deselected (parameter OTG_DFIFO_DYNAMIC = 0), these flops are optimized, and reads return the Power-on value. If Enable Dynamic FIFO Sizing? was selected (parameter OTG_DFIFO_DYNAMIC = 1), you can write a new value in this field. Programmed values must not exceed the Power-on value .</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	INEP1TxFStAddr IN Endpoint FIFO1 Transmit RAM Start Address This field contains the memory start address for IN endpoint Transmit FIFOOn (0 < n <= 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth (parameter OTG_RX_DFIFO_DEPTH). OTG_RX_DFIFO_DEPTH + SUM 0 to n-1 (OTG_DINEP_TXFIFO_DEPTH_n) For example start address of IN endpoint FIFO 1 is OTG_RX_DFIFO_DEPTH + OTG_DINEP_TXFIFO_DEPTH_0. The start address of IN endpoint FIFO 2 is OTG_RX_DFIFO_DEPTH + OTG_DINEP_TXFIFO_DEPTH_0 + OTG_DINEP_TXFIFO_DEPTH_1. If Enable Dynamic FIFO Sizing? was deselected (parameter OTG_DFIFO_DYNAMIC = 0), these flops are optimized, and reads return the power-on value. If Enable Dynamic FIFO Sizing? was selected (parameter OTG_DFIFO_DYNAMIC = 1), and you have programmed a new value for Rx FIFO depth, you can write that value in this field. Programmed values must not exceed the power-on value set .

**USBOTG\_HCFG**

Address: Operational Base + offset (0x0400)

Host Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:27	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
26	RW	0x0	PerSchedEna Enable Periodic Scheduling Applicable in Scatter/Gather DMA mode only. Enables periodic scheduling within the core. Initially, the bit is reset. The core will not process any periodic channels. As soon as this bit is set, the core will get ready to start scheduling periodic channels and sets HCFG.PerSchedStat. The setting of HCFG.PerSchedStat indicates the core has enabled periodic scheduling. Once HCFG.PerSchedEna is set, the application is not supposed to again reset the bit unless HCFG.PerSchedStat is set. As soon as this bit is reset, the core will get ready to stop scheduling periodic channels and resets HCFG.PerSchedStat. In non-Scatter/Gather DMA mode, this bit is reserved.
25:24	RW	0x0	FrListEn Frame List Entries The value in the register specifies the number of entries in the Frame list. This field is valid only in Scatter/Gather DMA mode.
23	RW	0x0	DescDMA Enable Scatter/gather DMA in Host mode When the Scatter/Gather DMA option selected during configuration of the RTL, the application can set this bit during initialization to enable the Scatter/Gather DMA operation. NOTE: This bit must be modified only once after a reset. The following combinations are available for programming: GAHBCFG.DMAEn=0, HCFG.DescDMA=0 => Slave mode GAHBCFG.DMAEn=0, HCFG.DescDMA=1 => Invalid GAHBCFG.DMAEn=1, HCFG.DescDMA=0 => Buffered DMA mode GAHBCFG.DMAEn=1, HCFG.DescDMA=1 => Scatter/Gather DMA mode In non-Scatter/Gather DMA mode, this bit is reserved.
22:16	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:8	RW	0x00	ResValid Resume Validation Period This field is effective only when HCFG.Ena32KHzS is set. It controls the resume period when the core resumes from suspend. The core counts the ResValid number of clock cycles to detect a valid resume when this is set.
7	RW	0x0	Ena32KHzS Enable 32-KHz Suspend Mode This bit can only be set if the USB 1.1 Full-Speed Serial Transceiver Interface has been selected. If USB 1.1 Full-Speed Serial Transceiver Interface has not been selected, this bit must be zero. When the USB 1.1 Full-Speed Serial Transceiver Interface is chosen and this bit is set, the core expects the 48-MHz PHY clock to be switched to 32 KHz during a suspend.
6:3	RO	0x0	reserved
2	RW	0x0	FSLSSupp FS- and LS-Only Support The application uses this bit to control the core enumeration speed. Using this bit, the application can make the core enumerate as a FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming. 1'b0: HS/FS/LS, based on the maximum speed supported by the connected device 1'b1: FS/LS-only, even if the connected device can support HS
1:0	RW	0x0	FSLSPclkSel FS/LS PHY Clock Select 2'b00: PHY clock is running at 30/60 MHz 2'b01: PHY clock is running at 48 MHz Others: Reserved

**USBOTG\_HFIR**

Address: Operational Base + offset (0x0404)

Host Frame Interval Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	<p>FrInt Frame Interval The value that the application programs to this field specifies the interval between two consecutive SOFs (FS) or micro-SOFs (HS) or Keep-Alive tokens (HS). This field contains the number of PHY clocks that constitute the required frame interval. The default value set in this field for a FS operation when the PHY clock frequency is 60 MHz. The application can write a value to this register only after the Port Enable bit of the Host Port Control and Status register (HPRT.PrtEnaPort) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY Clock Select field of the Host Configuration register (HCFG.FSLSPclkSel). Do not change the value of this field after the initial configuration.</p> <p>125 us * (PHY clock frequency for HS) 1 ms * (PHY clock frequency for FS/LS)</p>

**USBOTG\_HFNUM**

Address: Operational Base + offset (0x0408)

Host Frame Number/Frame Time Remaining Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	<p>FrRem Frame Time Remaining Indicates the amount of time remaining in the current micro-frame (HS) or frame (FS/LS), in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame Interval register and a new SOF is transmitted on the USB.</p>
15:0	RO	0xffff	<p>FrNum Frame Number This field increments when a new SOF is transmitted on the USB, and is reset to 0 when it reaches 16'h3FFF. This field is writable only if Remove Optional Features? was not selected (OTG_RM_OTG_FEATURES = 0). Otherwise, reads return the frame number value.</p>

**USBOTG\_HPTXSTS**

Address: Operational Base + offset (0x0410)

Host Periodic Transmit FIFO/Queue Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x00	<p>PTxQTop Top of the Periodic Transmit Request Queue This indicates the entry in the Periodic Tx Request Queue that is currently being processed by the MAC. This register is used for debugging.</p> <p>Bit [31]: Odd/Even (micro)frame 1'b0: send in even (micro)frame 1'b1: send in odd (micro)frame Bits [30:27]: Channel/endpoint number Bits [26:25]: Type 2'b00: IN/OUT 2'b01: Zero-length packet 2'b10: CSPLIT 2'b11: Disable channel command Bit [24]: Terminate (last entry for the selected channel/endpoint)</p>
23:16	RO	0x00	<p>PTxQSpAvail Periodic Transmit Request Queue Space Available Indicates the number of free locations available to be written in the Periodic Transmit Request Queue. This queue holds both IN and OUT requests.</p> <p>8'h0: Periodic Transmit Request Queue is full 8'h1: 1 location available 8'h2: 2 locations available n: n locations available (0 &lt;=n &lt;= 16) Others: Reserved</p>
15:0	RW	0x0000	<p>PTxFSpAvail Periodic Transmit Data FIFO Space Available Indicates the number of free locations available to be written to in the Periodic TxFIFO. Values are in terms of 32-bit words</p> <ul style="list-style-type: none"> <li>. 16'h0: Periodic TxFIFO is full</li> <li>. 16'h1: 1 word available</li> <li>. 16'h2: 2 words available</li> <li>. 16'hn: n words available (where 0 . n . 32,768)</li> <li>. 16'h8000: 32,768 words available</li> <li>. Others: Reserved</li> </ul>

**USBOTG\_HAIT**

Address: Operational Base + offset (0x0414)

Host All Channels Interrupt Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RO	0x0000	HAI NT Channel Interrupts One bit per channel: Bit 0 for Channel 0 Bit 1 for Channel 1 ..... Bit 15 for Channel 15

**USBOTG\_HINTMSK**

Address: Operational Base + offset (0x0418)

Host All Channels Interrupt Mask Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	HAI NTMsk Channel Interrupt Mask One bit per channel: Bit 0 for channel 0, bit 15 for channel 15

**USBOTG\_HPRT**

Address: Operational Base + offset (0x0440)

Host Port Control and Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:19	RO	0x0	reserved
18:17	RO	0x0	PrtSpd Port Speed Indicates the speed of the device attached to this port. 2'b00: High speed 2'b01: Full speed 2'b10: Low speed 2'b11: Reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16:13	RW	0x0	<p>PrtTstCtl Port Test Control The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.</p> <p>4'b0000: Test mode disabled 4'b0001: Test_J mode 4'b0010: Test_K mode 4'b0011: Test_SE0_NAK mode 4'b0100: Test_Packet mode 4'b0101: Test_Force_Enable Others: Reserved</p>
12	RWSC	0x0	<p>PrtPwr Port Power The application uses this field to control power to this port (write 1'b1 to set to 1'b1 and write 1'b0 to set to 1'b0), and the core can clear this bit on an over current condition.</p> <p>1'b0: Power off 1'b1: Power on</p>
11:10	RO	0x0	<p>PrtLnSts Port Line Status Indicates the current logic level USB data lines</p> <p>Bit [10]: Logic level of D+ Bit [11]: Logic level of D</p>
9	RO	0x0	reserved
8	RW	0x0	<p>PrtRst Port Reset When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.</p> <p>1'b0: Port not in reset 1'b1: Port in reset</p> <p>To start a reset on the port, the application must leave this bit set for at least the minimum duration mentioned below, as specified in the USB 2.0 specification, Section 7.1.7.5. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.</p> <p>High speed: 50 ms Full speed/Low speed: 10 ms</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RWSC	0x0	<p>PrtSusp Port Suspend</p> <p>The application sets this bit to put this port in Suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the Port Clock Stop bit, which asserts the suspended input pin of the PHY.</p> <p>The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wakeup signal is detected or the application sets the Port Reset bit or Port Resume bit in this register or the Resume/Remote Wakeup Detected Interrupt bit or Disconnect Detected Interrupt bit in the Core Interrupt register (GINTSTS.WkUpInt or GINTSTS.DisconnInt, respectively).</p> <p>1'b0: Port not in Suspend mode 1'b1: Port in Suspend mode</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RWSC	0x0	<p>PrtRes Port Resume The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.</p> <p>If the core detects a USB remote wakeup sequence, as indicated by the Port Resume/Remote Wakeup Detected Interrupt bit of the Core Interrupt register (GINTSTS.WkUpInt), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.</p> <p>1'b0: No resume driven 1'b1: Resume driven</p> <p>When LPM is enabled and the core is in the L1 (Sleep) state, setting this bit results in the following behavior:</p> <p>The core continues to drive the resume signal until a pre-determined time specified in the GLPMCFG.HIRD_Thres[3:0] field.</p> <p>If the core detects a USB remote wakeup sequence, as indicated by the Port L1 Resume/Remote L1 Wakeup Detected Interrupt bit of the Core Interrupt register (GINTSTS.L1WkUpInt), the core starts driving resume signaling without application intervention and clears this bit at the end of the resume. The read value of this bit indicates whether the core is currently driving resume signaling.</p> <p>1'b0: No resume driven 1'b1: Resume driven</p>
5	W1C	0x0	<p>PrtOvrCurrChng Port Overcurrent Change The core sets this bit when the status of the Port Overcurrent Active bit (bit 4) in this register changes.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RO	0x0	PrtOvrCurrAct Port Overcurrent Active Indicates the overcurrent condition of the port. 1'b0: No overcurrent condition 1'b1: Overcurrent condition
3	W1C	0x0	PrtEnChng Port Enable/Disable Change The core sets this bit when the status of the Port Enable bit [2] of this register changes.
2	W1C	0x0	PrtEna Port Enable A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application. 1'b0: Port disabled 1'b1: Port enabled
1	W1C	0x0	PrtConnDet Port Connect Detected The core sets this bit when a device connection is detected to trigger an interrupt to the application using the Host Port Interrupt bit of the Core Interrupt register (GINTSTS.PrtInt). The application must write a 1 to this bit to clear the interrupt.
0	RO	0x0	PrtConnSts Port Connect Status 1'b0: No device is attached to the port. 1'b1: A device is attached to the port.

**USBOTG\_HCCHARn**

Address: Operational Base + offset (0x0500)

Host Channel-n Characteristics Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RWSC	0x0	<p>ChEna Channel Enable When Scatter/Gather mode is enabled 1'b0: Indicates that the descriptor structure is not yet ready. 1'b1: Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor. When Scatter/Gather mode is disabled, This field is set by the application and cleared by the OTG host. 1'b0: Channel disabled 1'b1: Channel enabled</p>
30	RWSC	0x0	<p>ChDis Channel Disable The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled.</p>
29	RW	0x0	<p>OddFrm Odd Frame This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd (micro) frame. This field is applicable for only periodic (isochronous and interrupt) transactions. 1'b0: Even (micro)frame 1'b1: Odd (micro)frame This field is not applicable for Scatter/Gather DMA mode and need not be programmed by the application and is ignored by the core.</p>
28:22	RW	0x00	<p>DevAddr Device Address This field selects the specific device serving as the data source or sink.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
21:20	RW	0x0	<p>MC_EC Multi Count (MC) / Error Count (EC) When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.Spltna) is reset (1'b0), this field indicates to the host the number of transactions that must be executed per micro-frame for this periodic endpoint. For non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration.</p> <p>2'b00: Reserved This field yields undefined results. 2'b01: 1 transaction 2'b10: 2 transactions to be issued for this endpoint per micro-frame 2'b11: 3 transactions to be issued for this endpoint per micro-frame</p> <p>When HCSPLTn.Spltna is set (1'b1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 2'b01.</p>
19:18	RW	0x0	<p>EPType Endpoint Type Indicates the transfer type selected.</p> <p>2'b00: Control 2'b01: Isochronous 2'b10: Bulk 2'b11: Interrupt</p>
17	RW	0x0	<p>LSpdDev Low-Speed Device This field is set by the application to indicate that this channel is communicating to a low-speed device.</p>
16	RO	0x0	reserved
15	RW	0x0	<p>EPDir Endpoint Direction Indicates whether the transaction is IN or OUT.</p> <p>1'b0: OUT 1'b1: IN</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14:11	RW	0x0	EPNum Endpoint Number Indicates the endpoint number on the device serving as the data source or sink.
10:0	RW	0x000	MPS Maximum Packet Size Indicates the maximum packet size of the associated endpoint.

**USBOTG\_HCSPLTn**

Address: Operational Base + offset (0x0504)

Host Channel-n Split Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	SpltEna Split Enable The application sets this field to indicate that this channel is enabled to perform split transactions.
30:17	RO	0x0	reserved
16	RW	0x0	CompSplt Do Complete Split The application sets this field to request the OTG host to perform a complete split transaction.
15:14	RW	0x0	XactPos Transaction Position This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction.  2'b11: All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes). 2'b10: Begin. This is the first data payload of this transaction (which is larger than 188 bytes). 2'b00: Mid. This is the middle payload of this transaction (which is larger than 188bytes). 2'b01: End. This is the last payload of this transaction (which is larger than 188 bytes).
13:7	RW	0x00	HubAddr Hub Address This field holds the device address of the transaction translator's hub.
6:1	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	PrtAddr Port Address This field is the port number of the recipient transaction translator.

**USBOTG\_HCINTn**

Address: Operational Base + offset (0x0508)

Host Channel-n Interrupt Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13	W1C	0x0	DESC_LST_ROLLIntr Descriptor rollover interrupt This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. For non-Scatter/Gather DMA mode, this bit is reserved.
12	W1C	0x0	XCS_XACT_ERR Excessive Transaction Error This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels. For non-Scatter/Gather DMA mode, this bit is reserved.
11	W1C	0x0	BNAIntr BNA (Buffer Not Available) Interrupt This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. For non-Scatter/Gather DMA mode, this bit is reserved.
10	W1C	0x0	DataTglErr Data Toggle Error In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
9	W1C	0x0	FrmOvrn Frame Overrun In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	W1C	0x0	BblErr Babble Error In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
7	W1C	0x0	XactErr Transaction Error Indicates one of the following errors occurred on the USB: CRC check failure, Timeout, Bit stuff error, False EOP. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
6	WO	0x0	NYET NYET Response Received Interrupt In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
5	W1C	0x0	ACK ACK Response Received/Transmitted Interrupt In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
4	W1C	0x0	NAK NAK Response Received Interrupt In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
3	W1C	0x0	STALL STALL Response Received Interrupt In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
2	W1C	0x0	AHBErr AHB Error This is generated only in DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	W1C	0x0	ChHltd Channel Halted In non-Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/Gather DMA mode, this indicates that transfer completed due to any of the following: EOL being set in descriptor, AHB error, Excessive transaction errors, In response to disable request by the application, Babble, Stall, Buffer Not Available (BNA)
0	W1C	0x0	XferCompl Transfer Completed For Scatter/Gather DMA mode, it indicates that current descriptor processing got completed with IOC bit set in its descriptor. In non-Scatter/Gather DMA mode, it indicates that Transfer completed normally without any errors.

**USBOTG\_HCINTMSKn**

Address: Operational Base + offset (0x050c)

Host Channel-n Interrupt Mask Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13	RW	0x0	DESC_LST_ROLLIntrMsk Descriptor rollover interrupt Mask register This bit is valid only when Scatter/Gather DMA mode is enabled. In non-Scatter/Gather DMA mode, this bit is reserved.
12	RO	0x0	reserved
11	RW	0x0	BNAIntrMsk BNA (Buffer Not Available) Interrupt mask register This bit is valid only when Scatter/Gather DMA mode is enabled. In non-Scatter/Gather DMA mode, this bit is reserved.
10	RW	0x0	DataTglErrMsk Data Toggle Error Mask This bit is not applicable in Scatter/Gather DMA mode.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	FrmOvrnMsk Frame Overrun Mask This bit is not applicable in Scatter/Gather DMA mode.
8	RW	0x0	BblErrMsk Babble Error Mask This bit is not applicable in Scatter/Gather DMA mode.
7	RW	0x0	XactErrMsk Transaction Error Mask This bit is not applicable in Scatter/Gather DMA mode
6	RW	0x0	NyetMsk NYET Response Received Interrupt Mask This bit is not applicable in Scatter/Gather DMA mode.
5	RW	0x0	AckMsk ACK Response Received/Transmitted Interrupt Mask This bit is not applicable in Scatter/Gather DMA mode.
4	RW	0x0	NakMsk NAK Response Received Interrupt Mask This bit is not applicable in Scatter/Gather DMA mode.
3	RW	0x0	StallMsk STALL Response Received Interrupt Mask This bit is not applicable in Scatter/Gather DMA mode.
2	RW	0x0	AHBErrMsk AHB Error Mask Note: This bit is only accessible when OTG_ARCHITECTURE = 2
1	RW	0x0	ChHltdMsk Channel Halted Mask
0	RW	0x0	XferComplMsk Transfer Completed Mask This bit is valid only when Scatter/Gather DMA mode is enabled. In non-Scatter/Gather DMA mode, this bit is reserved.

**USBOTG\_HCTSIZn**

Address: Operational Base + offset (0x0510)

Host Channel-n Transfer Size Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	<p>DoPng Do Ping</p> <p>This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol. Note: Do not set this bit for IN transfers. If this bit is set for IN transfers it disables the channel.</p>
30:29	RW	0x0	<p>Pid PID</p> <p>The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.</p> <p>2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA (non-control)/SETUP (control)</p>
28:19	RW	0x000	<p>PktCnt Packet Count</p> <p>This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as Width of Packet Counters (parameter OTG_PACKET_COUNT_WIDTH).</p>
18:0	RW	0x00000	<p>XferSize Transfer Size</p> <p>For an OUT, this field is the number of data bytes the host sends during the transfer. For an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic). The width of this counter is specified as Width of Transfer Size Counters (parameter OTG_TRANS_COUNT_WIDTH).</p>

**USBOTG\_HCDMAN**

Address: Operational Base + offset (0x0514)  
Host Channel-n DMA Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	DMAAddr DMA Address This field holds the start address in the external memory from which the data for the endpoint must be fetched or to which it must be stored. This register is incremented on every AHB transaction.

**USBOTG\_HCDMABn**

Address: Operational Base + offset (0x051c)

Host Channel-n DMA Buffer Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HCDMABn Holds the current buffer address This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

**USBOTG\_DCFG**

Address: Operational Base + offset (0x0800)

Device Configuration Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RW	0x02	ResValid Resume Validation Period This field controls the period when the core resumes from a suspend. When this bit is set, the core counts for the ResValid number of clock cycles to detect a valid resume. This field is effective only when DCFG.Ena32KHzSusp is set.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
25:24	RW	0x0	<p>PerSchIntvl Periodic Scheduling Interval PerSchIntvl must be programmed only for Scatter/Gather DMA mode. Description: This field specifies the amount of time the Internal DMA engine must allocate for fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25,50 or 75% of (micro)frame. When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data. When no periodic endpoints are active, then the internal DMA engine services nonperiodic endpoints, ignoring this field. After the specified time within a (micro) frame, the DMA switches to fetching for nonperiodic endpoints.</p> <p>2'b00: 25% of (micro) frame. 2'b01: 50% of (micro) frame. 2'b10: 75% of (micro) frame. 2'b11: Reserved.</p>
23	RW	0x0	<p>DescDMA Enable Scatter/Gather DMA in Device mode When the Scatter/Gather DMA option selected during configuration of the RTL, the application can set this bit during initialization to enable the Scatter/Gather DMA operation. NOTE: This bit must be modified only once after a reset. The following combinations are available for programming:</p> <ul style="list-style-type: none"> <li>GAHBCFG.DMAEn=0, DCFG.DescDMA=0 =&gt; Slave mode</li> <li>GAHBCFG.DMAEn=0, DCFG.DescDMA=1 =&gt; Invalid</li> <li>GAHBCFG.DMAEn=1, DCFG.DescDMA=0 =&gt; Buffered DMA mode</li> <li>GAHBCFG.DMAEn=1, DCFG.DescDMA=1 =&gt; Scatter/Gather DMA mode</li> </ul>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
22:18	RW	0x08	EPMisCnt IN Endpoint Mismatch Count This field is valid only in shared FIFO operation. The application programs this field with a count that determines when the core generates an Endpoint Mismatch interrupt (GINTSTS.EPMis). The core loads this value into an internal counter and decrements it. The counter is reloaded whenever there is a match or when the counter expires. The width of this counter depends on the depth of the Token Queue.
17:13	RO	0x0	reserved
12:11	RW	0x0	PerFrInt Periodic Frame Interval Indicates the time within a (micro) frame at which the application must be notified using the End Of Periodic Frame Interrupt. This can be used to determine if all the isochronous traffic for that (micro) frame is complete. 2'b00: 80% of the (micro)frame interval 2'b01: 85% 2'b10: 90% 2'b11: 95%
10:4	RW	0x00	DevAddr Device Address The application must program this field after every SetAddress control command.
3	RW	0x0	Ena32KHzS Enable 32-KHz Suspend Mode When the USB 1.1 Full-Speed Serial Transceiver Interface is chosen and this bit is set, the core expects the 48-MHz PHY clock to be switched to 32 KHz during a suspend. This bit can only be set if USB 1.1 Full-Speed Serial Transceiver Interface has been selected. If USB 1.1 Full-Speed Serial Transceiver Interface has not been selected, this bit must be zero.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x0	<p>NZStsOUTHShk Non-Zero-Length Status OUT Handshake The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's Status stage.</p> <p>1'b1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.</p> <p>1'b0: Send the received OUT packet to the application (zero-length or nonzerolength) and send a handshake based on the NAK and STALL bits for the endpoint in the Device Endpoint Control register.</p>
1:0	RW	0x0	<p>DevSpd Device Speed Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.</p> <p>2'b00: High speed (USB 2.0 PHY clock is 30 MHz or 60 MHz)</p> <p>2'b01: Full speed (USB 2.0 PHY clock is 30 MHz or 60 MHz)</p> <p>2'b10: Reserved</p> <p>2'b11: Full speed (USB 1.1 transceiver clock is 48 MHz)</p>

**USBOTG\_DCTL**

Address: Operational Base + offset (0x0804)

Device Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved
16	RW	0x0	<p>NakOnBble Set NAK automatically on babble The core sets NAK automatically for the endpoint on which babble is received.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RW	0x0	<p>IgnrFrmNum Ignore frame number for isochronous endpoints in case of Scatter/</p> <p>Do NOT program IgnrFrmNum bit to 1'b1 when the core is operating in Threshold mode.</p> <p>Note: When Scatter/Gather DMA mode is enabled this feature is not applicable to highspeed, high-bandwidth transfers. When this bit is enabled, there must be only one packet per descriptor.</p> <p>1'b0: The core transmits the packets only in the frame number in which they are intended to be transmitted.</p> <p>1'b1: The core ignores the frame number, sending packets immediately as the packets are ready.</p> <p>Scatter/Gather: In Scatter/Gather DMA mode, when this bit is enabled, the packets are not flushed when an ISOC IN token is received for an elapsed frame.</p> <p>When Scatter/Gather DMA mode is disabled, this field is used by the application to enable periodic transfer interrupt. The application can program periodic endpoint transfers for multiple (micro) frames.</p> <p>1'b0: Periodic transfer interrupt feature is disabled; the application must program transfers for periodic endpoints every (micro)frame</p> <p>1'b1: Periodic transfer interrupt feature is enabled; the application can program transfers for multiple (micro)frames for periodic endpoints.</p> <p>In non-Scatter/Gather DMA mode, the application receives transfer complete interrupt after transfers for multiple (micro) frames are completed.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14:13	RW	0x1	GMC Global Multi Count GMC must be programmed only once after initialization. Applicable only for Scatter/Gather DMA mode. This indicates the number of packets to be serviced for that end point before moving to the next end point. It is only for nonperiodic end points. 2'b00: Invalid. 2'b01: 1 packet. 2'b10: 2 packets. 2'b11: 3 packets. When Scatter/Gather DMA mode is disabled, this field is reserved. and reads 2'b00.
12	RO	0x0	reserved
11	RW	0x0	PWROnPrgDone Power-On Programming Done The application uses this bit to indicate that register programming is completed after a wake-up from Power Down mode.
10	WO	0x0	CGOUTNak Clear Global OUT NAK A write to this field clears the Global OUT NAK.
9	WO	0x0	SGOUTNak Set Global OUT NAK A write to this field sets the Global OUT NAK. The application uses this bit to send a NAK handshake on all OUT endpoints. The application must set this bit only after making sure that the Global OUT NAK Effective bit in the Core Interrupt Register (GINTSTS.GOUTNakEff) is cleared.
8	WO	0x0	CGNPInNak Clear Global Non-periodic IN NAK A write to this field clears the Global Non-periodic IN NAK.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	WO	0x0	SGNPIInNak Set Global Non-periodic IN NAK A write to this field sets the Global Non-periodic IN NAK. The application uses this bit to send a NAK handshake on all non-periodic IN endpoints. The core can also set this bit when a timeout condition is detected on a non-periodic endpoint in shared FIFO operation. The application must set this bit only after making sure that the Global IN NAK Effective bit in the Core Interrupt Register (GINTSTS.GINNakEff) is cleared.
6:4	RW	0x0	TstCtl Test Control 3'b000: Test mode disabled 3'b001: Test_J mode 3'b010: Test_K mode 3'b011: Test_SE0_NAK mode 3'b100: Test_Packet mode 3'b101: Test_Force_Enable Others: Reserved
3	RO	0x0	GOUTNakSts Global OUT NAK Status 1'b0: A handshake is sent based on the FIFO Status and the NAK and STALL bit settings. 1'b1: No data is written to the RxFIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped
2	RO	0x0	GNPINNakSts Global Non-periodic IN NAK Status 1'b0: A handshake is sent out based on the data availability in the transmit FIFO. 1'b1: A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	<p>SftDiscon Soft Disconnect The application uses this bit to signal the Otg core to do a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.</p> <p>1'b0: Normal operation. When this bit is cleared after a soft disconnect, the core drives the phy_opmode_o signal on the UTMI+ to 2'b00, which generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.</p> <p>1'b1: The core drives the phy_opmode_o signal on the UTMI+ to 2'b01, which generates a device disconnect event to the USB host.</p>
0	RW	0x0	<p>RmtWkUpSig Remote Wakeup Signaling When the application sets this bit, the core initiates remote signaling to wake the USB host. The application must set this bit to instruct the core to exit the Suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1-15 ms after setting it. If LPM is enabled and the core is in the L1 (Sleep) state, when the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the Sleep state. As specified in the LPM specification, the hardware automatically clears this bit 50 us (TL1DevDrvResume) after being set by the application. The application must not set this bit when GLPMCFG.bRemoteWake from the previous LPM transaction is zero.</p>

**USBOTG\_DSTS**

Address: Operational Base + offset (0x0808)

Device Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RO	0x0	reserved
21:8	RW	0x0000	<p>SOFFN Frame or Micro-frame Number of the Received SOF</p> <p>When the core is operating at high speed, this field contains a micro-frame number. When the core is operating at full or low speed, this field contains a frame number.</p>
7:4	RO	0x0	reserved
3	RW	0x0	<p>ErrticErr Erratic Error</p> <p>The core sets this bit to report any erratic errors (phy_rxvalid_i/phy_rxvldh_i or phy_rxactive_i is asserted for at least 2 ms, due to PHY error) seen on the UTMI+. Due to erratic errors, the Otg core goes into Suspended state and an interrupt is generated to the application with Early Suspend bit of the Core Interrupt register (GINTSTS.ErlySusp). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.</p>
2:1	RW	0x0	<p>EnumSpd Enumerated Speed</p> <p>Indicates the speed at which the OTG core has come up after speed detection through a chirp sequence.</p> <p>2'b00: High speed (PHY clock is running at 30 or 60 MHz)</p> <p>2'b01: Full speed (PHY clock is running at 30 or 60 MHz)</p> <p>2'b10: Low speed (PHY clock is running at 48 MHz, internal phy_clk at 6 MHz)</p> <p>2'b11: Full speed (PHY clock is running at 48 MHz)</p> <p>Low speed is not supported for devices using a UTMI+ PHY.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	SuspSts Suspend Status In Device mode, this bit is set as long as a Suspend condition is detected on the USB. The core enters the Suspended state when there is no activity on the utmi_linestate signal for an extended period of time. The core comes out of the suspend: When there is any activity on the utmi_linestate signal, When the application writes to the Remote Wakeup Signaling bit in the Device Control register (DCTL.RmtWkUpSig).

**USBOTG\_DIEPMSK**

Address: Operational Base + offset (0x0810)

Device IN Endpoint common interrupt mask register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13	RW	0x0	NAKMsk NAK interrupt Mask
12:10	RO	0x0	reserved
9	RW	0x0	BNAInIntrMsk BNA Interrupt Mask
8	RW	0x0	TxfifoUndrnMsk Fifo Underrun Mask
7	RO	0x0	reserved
6	RW	0x0	INEPNakEffMsk IN Endpoint NAK Effective Mask
5	RW	0x0	INTknEPMisMsk IN Token received with EP Mismatch Mask
4	RW	0x0	INTknTxFTEmpMsk IN Token Received When TxFIFO Empty Mask
3	RW	0x0	TimeOUTMsk Timeout Condition Mask
2	RW	0x0	AHBErrMsk AHB Error Mask
1	RW	0x0	EPDisblIdMsk Endpoint Disabled Interrupt Mask
0	RW	0x0	XferComplMsk Transfer Completed Interrupt Mask

**USBOTG\_DOEPMSK**

Address: Operational Base + offset (0x0814)

Device OUT Endpoint common interrupt mask register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	RO	0x0	reserved
14	RW	0x0	NYETMsk NYET Interrupt Mask
13	RW	0x0	NAKMsk NAK Interrupt Mask
12	RW	0x0	BbleErrMsk Babble Interrupt Mask
11:10	RO	0x0	reserved
9	RW	0x0	BnaOutIntrMsk BNA interrupt Mask
8	RW	0x0	OutPktErrMsk OUT Packet Error Mask
7	RO	0x0	reserved
6	RW	0x0	Back2BackSETup Back-to-Back SETUP Packets Received Mask Applies to control OUT endpoints only.
5	RO	0x0	reserved
4	RW	0x0	OUTTknEPdisMsk OUT Token Received when Endpoint Disabled Mask Applies to control OUT endpoints only.
3	RW	0x0	SetUPMsk SETUP Phase Done Mask Applies to control endpoints only.
2	RW	0x0	AHBErrMsk AHB Error
1	RW	0x0	EPDisbldMsk Endpoint Disabled Interrupt Mask
0	RW	0x0	XferComplMsk Transfer Completed Interrupt Mask

**USBOTG\_DAIT**

Address: Operational Base + offset (0x0818)

Device All Endpoints interrupt register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	OutEPInt OUT Endpoint Interrupt Bits One bit per OUT endpoint: Bit 16 for OUT endpoint 0, bit 31 for OUT endpoint 15
15:0	RO	0x0000	InEpInt IN Endpoint Interrupt Bits One bit per IN Endpoint: Bit 0 for IN endpoint 0, bit 15 for endpoint 15

**USBOTG\_DAINTMASK**

Address: Operational Base + offset (0x081c)

Device All Endpoint interrupt mask register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	OutEpMsk OUT EP Interrupt Mask Bits One per OUT Endpoint: Bit 16 for OUT EP 0, bit 31 for OUT EP 15
15:0	RW	0x0000	InEpMsk IN EP Interrupt Mask Bits One bit per IN Endpoint: Bit 0 for IN EP 0, bit 15 for IN EP 15

**USBOTG\_DTKNQR1**

Address: Operational Base + offset (0x0820)

Device IN token sequence learning queue read register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0000000	EPTkn Endpoint Token Four bits per token represent the endpoint number of the token: Bits [31:28]: Endpoint number of Token 5 Bits [27:24]: Endpoint number of Token 4 ..... Bits [15:12]: Endpoint number of Token 1 Bits [11:8]: Endpoint number of Token 0
7	RO	0x0	WrapBit Wrap Bit This bit is set when the write pointer wraps. It is cleared when the learning queue is cleared.
6:5	RO	0x0	reserved
4:0	RO	0x00	INTknWPtr IN Token Queue Write Pointer

**USBOTG\_DTKNQR2**

Address: Operational Base + offset (0x0824)

Device IN token sequence learning queue read register2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	EPTkn Endpoint Token Four bits per token represent the endpoint number of the token: Bits [31:28]: Endpoint number of Token 13 Bits [27:24]: Endpoint number of Token 12 ..... Bits [7:4]: Endpoint number of Token 7 Bits [3:0]: Endpoint number of Token 6

**USBOTG\_DVBUSDIS**

Address: Operational Base + offset (0x0828)

Device VBUS discharge time register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0b8f	DVBUSDis Device VBUS Discharge Time Specifies the VBUS discharge time after VBUS pulsing during SRP. This value equals: VBUS discharge time in PHY clocks / 1,024. The value you use depends whether the PHY is operating at 30 MHz (16-bit data width) or 60 MHz (8-bit data width). Depending on your VBUS load, this value can need adjustment.

**USBOTG\_DVBUSPULSE**

Address: Operational Base + offset (0x082c)

Device VBUS Pulsing Timer Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	DVBUSPulse Device VBUS Pulsing Time Specifies the VBUS pulsing time during SRP. This value equals: VBUS pulsing time in PHY clocks / 1,024. The value you use depends whether the PHY is operating at 30 MHz (16-bit data width) or 60 MHz (8-bit data width).

**USBOTG\_DTHRCTL**

Address: Operational Base + offset (0x0830)

Device Threshold Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27	RW	0x1	ArbPrkEn Arbiter Parking Enable This bit controls internal DMA arbiter parking for IN endpoints. When threshold is enabled and this bit is set to one, then the arbiter parks on the IN endpoint for which there is a token received on the USB. This is done to avoid getting into under-run conditions. By default the parking is enabled.
26	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
25:17	RW	0x008	<p>RxThrLen Receive Threshold Length This field specifies Receive threshold size in DWORDS. This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight DWORDS.</p> <p>The recommended value for ThrLen is to be the same as the programmed AHB Burst Length (GAHBCFG.HBstLen).</p>
16	RW	0x0	<p>RxThrEn Receive Threshold Enable When this bit is set, the core enables thresholding in the receive direction.</p>
15:13	RO	0x0	reserved
12:11	RW	0x0	<p>AHBThrRatio AHB Threshold Ratio These bits define the ratio between the AHB threshold and the MAC threshold for the transmit path only. The AHB threshold always remains less than or equal to the USB threshold, because this does not increase overhead. Both the AHB and the MAC threshold must be DWORD-aligned. The application needs to program TxThrLen and the AHBThrRatio to make the AHB Threshold value DWORD aligned. If the AHB threshold value is not WORD aligned, the core might not behave correctly. When programming the TxThrLen and AHBThrRatio, the application must ensure that the minimum AHB threshold value does not go below 8 DWORDS to meet the USB turnaround time requirements.</p> <p>2'b00: AHB threshold = MAC threshold 2'b01: AHB threshold = MAC threshold / 2 2'b10: AHB threshold = MAC threshold / 4 2'b11: AHB threshold = MAC threshold / 8</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
10:2	RW	0x008	<p>TxThrLen Transmit Threshold Length This field specifies Transmit threshold size in DWORDS. This field also forms the MAC threshold and specifies the amount of data, in bytes, to be in the corresponding endpoint transmit FIFO before the core can start a transaction on the USB. When the value of AHBThrRatio is 2'h00, the threshold length must be at least 8 DWORDS. If the AHBThrRatio is nonzero, the application must ensure that the AHB threshold value does not go below the recommended 8 DWORDs.</p> <p>This field controls both isochronous and non-isochronous IN endpoint thresholds. The recommended value for ThrLen is to be the same as the programmed AHB Burst Length (GAHBCFG.HBstLen).</p>
1	RW	0x0	<p>ISOThrEn ISO IN Endpoints Threshold Enable When this bit is set, the core enables threshold for isochronous IN endpoints.</p>
0	RW	0x0	<p>NonISOThrEn Non-ISO IN Endpoints Threshold Enable When this bit is set, the core enables threshold for Non Isochronous IN endpoints.</p>

**USBOTG\_DIEPEMPMSK**

Address: Operational Base + offset (0x0834)

Device IN endpoint FIFO empty interrupt mask register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	<p>InEpTxfEmpMsk IN EP Tx FIFO Empty Interrupt Mask Bits These bits act as mask bits for DIEPINTn. TxFEmp interrupt One bit per IN Endpoint: Bit 0 for IN endpoint 0 ... Bit 15 for endpoint 15</p>

**USBOTG\_DEACHINT**

Address: Operational Base + offset (0x0838)

Device each endpoint interrupt register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	EchOutEpInt OUT Endpoint Interrupt Bits One bit per OUT endpoint: Bit 16 for OUT endpoint 0 ... Bit 31 for OUT endpoint 15
15:0	RO	0x0000	EchInEpInt IN Endpoint Interrupt Bits One bit per IN Endpoint: Bit 0 for IN endpoint 0 ... Bit 15 for endpoint 15

**USBOTG\_DEACHINTMSK**

Address: Operational Base + offset (0x083c)

Device each endpoint interrupt register mask

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	EchOutEpMsk OUT EP Interrupt Mask Bits One per OUT Endpoint: Bit 16 for IN endpoint 0 ... Bit 31 for endpoint 15
15:0	RW	0x0000	EchInEpMsk IN EP Interrupt Mask Bits One bit per IN Endpoint: Bit 0 for IN endpoint 0 ... Bit 15 for endpoint 15

**USBOTG\_DIEPEACHMSKn**

Address: Operational Base + offset (0x0840)

Device each IN endpoint -n interrupt Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13	RW	0x0	NAKMsk NAK interrupt Mask
12:10	RO	0x0	reserved
9	RW	0x0	BNAInIntrMsk BNA interrupt Mask
8	RW	0x0	TxfifoUndrnMsk Fifo Under run Mask
7	RO	0x0	reserved
6	RW	0x0	INEPNakEffMsk IN Endpoint NAK Effective Mask

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	INTknEPMisMsk IN Token received with EP Mismatch Mask
4	RW	0x0	INTknTxFEmpMsk IN Token Received When TxFIFO Empty Mask
3	RW	0x0	TimeOUTMsk Timeout Condition Mask(Non-isochronous endpoints)
2	RW	0x0	AHBErrMsk AHB Error Mask
1	RW	0x0	EPDisblIdMsk Endpoint Disabled Interrupt Mask
0	RW	0x0	XferComplMsk Transfer Completed Interrupt Mask

**USBOTG\_DOEPEACHMSKn**

Address: Operational Base + offset (0x0880)

Device each out endpoint-n interrupt register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	RO	0x0	reserved
14	RW	0x0	NYETMsk NYET interrupt Mask
13	RW	0x0	NAKMsk NAK interrupt Mask
12	RW	0x0	BbleErrMsk Babble interrupt Mask
11:10	RO	0x0	reserved
9	RW	0x0	BnaOutIntrMsk BNA interrupt Mask
8	RW	0x0	OutPktErrMsk OUT Packet Error Mask
7	RO	0x0	reserved
6	RW	0x0	Back2BackSETup Back-to-Back SETUP Packets Received Mask Applies to control OUT endpoints only.
5	RO	0x0	reserved
4	RW	0x0	OUTTknEPdisMsk OUT Token Received when Endpoint Disabled Mask Applies to control OUT endpoints only.
3	RW	0x0	SetUPMsk SETUP Phase Done Mask Applies to control endpoints only.
2	RW	0x0	AHBErrMsk AHB Error

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	EPDisIdMsk Endpoint Disabled Interrupt Mask
0	RW	0x0	XferComplMsk Transfer Completed Interrupt Mask

**USBOTG\_DIEPCTL0**

Address: Operational Base + offset (0x0900)

Device control IN endpoint 0 control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RWSC	0x0	EPEna Endpoint Enable When Scatter/Gather DMA mode is enabled, for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. When Scatter/Gather DMA mode is disabled-such as in buffer-pointer based DMA mode-this bit indicates that data is ready to be transmitted on the endpoint. The core clears this bit before setting the following interrupts on this endpoint: Endpoint Disabled; Transfer Completed.
30	RWSC	0x0	EPDis Endpoint Disable The application sets this bit to stop transmitting data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled Interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint.
29:28	RO	0x0	reserved
27	WO	0x0	SNAK Set NAK A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for an endpoint after a SETUP packet is received on that endpoint.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
26	WO	0x0	CNAK Clear NAK A write to this bit clears the NAK bit for the endpoint.
25:23	RO	0x0	reserved
22	RW	0x0	TxFNum TxFIFO Number For Shared FIFO operation, this value is always set to 0, indicating that control IN endpoint 0 data is always written in the Non-Periodic Transmit FIFO. For Dedicated FIFO operation, this value is set to the FIFO number that is assigned to IN Endpoint 0.
21	RWSC	0x0	Stall STALL Handshake The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority.
20	RO	0x0	reserved
19:18	RO	0x0	EPType Endpoint Type Hardcoded to 00 for control
17	RO	0x0	NAKsts NAK Status Indicates the following: 1'b0: The core is transmitting non-NAK handshakes based on the FIFO status 1'b1: The core is transmitting NAK handshakes on this endpoint. When this bit is set, either by the application or core, the core stops transmitting data, even if there is data available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
16	RO	0x0	reserved
15	RO	0x1	USBActEP USB Active Endpoint This bit is always set to 1, indicating that control endpoint 0 is always active in all configurations and interfaces.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14:11	RW	0x0	NextEp Next Endpoint Applies to non-periodic IN endpoints only. Indicates the endpoint number to be fetched after the data for the current endpoint is fetched. The core can access this field, even when the Endpoint Enable (EPEna) bit is not set. This field is not valid in Slave mode. Note: This field is valid only for Shared FIFO operations.
10:2	RO	0x0	reserved
1:0	RW	0x0	MPS Maximum Packet Size Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. 2'b00: 64 bytes 2'b01: 32 bytes 2'b10: 16 bytes 2'b11: 8 bytes

**USBOTG\_DIEPINTn**

Address: Operational Base + offset (0x0908)

Device Endpoint-n Interrupt Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	RO	0x0	reserved
14	W1C	0x0	NYETInrpt NYET interrupt The core generates this interrupt when a NYET response is transmitted for a non-isochronous OUT endpoint.
13	W1C	0x0	NAKInrpt NAK interrupt The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.
12	W1C	0x0	BbleErrInrpt BbleErr (Babble Error) interrupt The core generates this interrupt when babble is received for the endpoint.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	W1C	0x0	PktDrpSts Packet Dropped Status This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non-Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.
10	RO	0x0	reserved
9	W1C	0x0	BNAIntr BNA (Buffer Not Available) Interrupt The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done Dependency: This bit is valid only when Scatter/Gather DMA mode is enabled.
8	W1C	0x0	TxfifoUndrn FIFO Under-run Applies to IN endpoints only. The core generates this interrupt when it detects a transmit FIFO under-run condition for this endpoint. Dependency: This interrupt is valid only when both of the following conditions are true: Parameter OTG_ENDED_TX_FIFO==1; Threshold is enabled; OUT Packet Error(OutPktErr). Applies to OUT endpoints only. This interrupt is asserted when the core detects an overflow or a CRC error for an OUT packet. Dependency: This interrupt is valid only when both of the following conditions are true: Parameter OTG_ENDED_TX_FIFO==1; Threshold is enabled.
7	W1C	0x0	TxFEmp Transmit FIFO Empty This bit is valid only for IN Endpoints. This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl).

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	W1C	0x0	<p>INEPNakEff IN Endpoint NAK Effective Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. This bit is applicable only when the endpoint is enabled.</p> <p>Back-to-Back SETUP Packets Received (Back2BackSETUp) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.</p>
5	W1C	0x0	<p>INTknEPMis IN Token Received with EP Mismatch Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.</p> <p>Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in case of Scatter Gather DMA mode.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	W1C	0x0	INTknTxFEmp IN Token Received When TxFIFO is Empty Indicates that an IN token was received when the associated TxFIFO periodic/nonperiodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. OUT Token Received When Endpoint Disabled (OUTTknEPdis) Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.
3	W1C	0x0	TimeOUT Timeout Condition In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.
2	W1C	0x0	AHBErr AHB Error Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.
1	W1C	0x0	EPDisbld Endpoint Disabled Interrupt Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	W1C	0x0	XferCompl Transfer Completed Interrupt Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled: For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

**USBOTG\_DIEPTSIZn**

Address: Operational Base + offset (0x0910)

Device endpoint n transfer size register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
30:29	RW	0x0	<p>MC Multi Count Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.</p> <p>2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets</p> <p>For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp). Received Data PID (RxDPID)</p> <p>Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint.</p> <p>2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA</p> <p>SETUP Packet Count (SUPCnt). Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive.</p> <p>2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
28:19	RW	0x000	PktCnt Packet Count Indicates the total number of USB packets that constitute the Transfer Size amount of data for this endpoint. The power-on value is specified for Width of Packet Counters during coreConsultant configuration (parameter OTG_PACKET_COUNT_WIDTH). IN Endpoints: This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO. OUT Endpoints: This field is decremented every time a packet (maximum size or short packet) is written to the RxFIFO.
18:0	RW	0x00000	XferSize Transfer Size This field contains the transfer size in bytes for the current endpoint. The power-on value is specified for Width of Transfer Size Counters during configuration (parameter OTG_TRANS_COUNT_WIDTH). The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. IN Endpoints: The core decrements this field every time a packet from the external memory is written to the TxFIFO. OUT Endpoints: The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.

**USBOTG\_DIEPDMAN**

Address: Operational Base + offset (0x0914)

Device endpoint-n DMA address register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	DMAAddr DMA Address Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.

**USBOTG\_DTXFSTS<sub>n</sub>**

Address: Operational Base + offset (0x0918)

Device IN endpoint transmit FIFO status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	INEPTxFSpAvail IN Endpoint TxFIFO Space Avail Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 . n . 32,768) 16'h8000: 32,768 words available Others: Reserved

**USBOTG\_DIEPDMA<sub>Bn</sub>**

Address: Operational Base + offset (0x091c)

Device endpoint-n DMA buffer address register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	DMABufferAddr DMA Buffer Address Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

**USBOTG\_DIEPCTLn**

Address: Operational Base + offset (0x0920)

Device endpoint-n control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RWSC	0x0	EPEna Endpoint Enable Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled, For IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. For OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. When Scatter/Gather DMA mode is enabled-such as for buffer-pointer based DMA mode: For IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint ; For OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. The core clears this bit before setting any of the following interrupts on this endpoint: SETUP Phase Done, Endpoint Disabled, Transfer Completed. Note: For control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
30	R/WSC	0x0	EPDis Endpoint Disable Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint.
29	WO	0x0	SetD1PID Set DATA1 PID Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr). Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro) frame (EO_FrNum) field to odd (micro) frame. This field is not applicable for Scatter/Gather DMA mode.
28	WO	0x0	SetD0PID Set DATA0 PID Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro) frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro) frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receipt descriptor structure.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27	WO	0x0	SNAK Set NAK Applies to IN and OUT endpoints. A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.
26	WO	0x0	CNAK Clear NAK Applies to IN and OUT endpoints. A write to this bit clears the NAK bit for the endpoint.
25:22	RW	0x0	TxFNum TxFIFO Number Shared FIFO Operation: non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO; Others: Specified Periodic TxFIFO number. Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation: these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
21	RW	0x0	<p>Stall STALL Handshake</p> <p>Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>
20	RW	0x0	<p>Snp Snoop Mode</p> <p>Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>
19:18	RW	0x0	<p>EPType Endpoint Type</p> <p>Applies to IN and OUT endpoints. This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control 2'b01: Isochronous 2'b10: Bulk 2'b11: Interrupt</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17	RO	0x0	<p>NAKSts NAK Status Applies to IN and OUT endpoints. Indicates the following:</p> <ul style="list-style-type: none"> <li>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</li> <li>1'b1: The core is transmitting NAK handshakes on this endpoint.</li> </ul> <p>When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16	RO	0x0	<p>DPID Endpoint Data PID Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetDOPID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode:</p> <p>Applies to isochronous IN and OUT endpoints only. Indicates the (micro) frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>
15	RWSC	0x0	<p>USBActEP USB Active Endpoint Applies to IN and OUT endpoints. Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14:11	RW	0x0	NextEp Next Endpoint Applies to non-periodic IN endpoints only. Indicates the endpoint number to be fetched after the data for the current endpoint is fetched. The core can access this field, even when the Endpoint Enable (EPEna) bit is low. This field is not valid in Slave mode operation. Note: This field is valid only for Shared FIFO operations.
10:0	RW	0x000	MPS Maximum Packet Size Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

**USBOTG\_DOEPCTL0**

Address: Operational Base + offset (0x0b00)

Device control OUT endpoint 0 control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RWSC	0x0	EPEna Endpoint Enable When Scatter/Gather DMA mode is enabled, for OUT endpoints this bit indicates that the descriptor structure and data buffer to receive data is setup. When Scatter/Gather DMA mode is disabled? such as for buffer-pointer based DMA mode)-this bit indicates that the application has allocated the memory to start receiving data from the USB. The core clears this bit before setting any of the following interrupts on this endpoint: SETUP Phase Done, Endpoint Disabled, Transfer Completed. Note: In DMA mode, this bit must be set for the core to transfer SETUP data packets into memory.
30	WO	0x0	EPDis Endpoint Disable The application cannot disable control OUT endpoint 0.
29:28	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27	WO	0x0	SNAK Set NAK A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set bit on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.
26	WO	0x0	CNAK Clear NAK A write to this bit clears the NAK bit for the endpoint.
25:22	RO	0x0	reserved
21	RWSC	0x0	Stall STALL Handshake The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
20	RW	0x0	Snp Snoop Mode This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.
19:18	RO	0x0	EPType Endpoint Type Hardcoded to 2'b00 for control.
17	RO	0x0	NAKsts NAK Status Indicates the following: 1'b0: The core is transmitting non-NAK handshakes based on the FIFO status. 1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit, the core stops receiving data, even if there is space in the RxFIFO to accommodate the incoming packet. Irrespective of this bit setting, the core always responds to SETUP data packets with an ACK handshake.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16	RO	0x0	reserved
15	RO	0x0	USBActEP USB Active Endpoint This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.
14:2	RO	0x0	reserved
1:0	RO	0x0	MPS Maximum Packet Size The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN Endpoint 0. 2'b00: 64 bytes 2'b01: 32 bytes 2'b10: 16 bytes 2'b11: 8 bytes

**USBOTG\_DOEPINTn**

Address: Operational Base + offset (0x0b08)

Device endpoint-n control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	RO	0x0	reserved
14	W1C	0x0	NYETIntrpt NYET interrupt The core generates this interrupt when a NYET response is transmitted for a non-isochronous OUT endpoint.
13	W1C	0x0	NAKIntrpt NAK interrupt The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.
12	W1C	0x0	BbleErrIntrpt BbleErr (Babble Error) interrupt The core generates this interrupt when babble is received for the endpoint.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	W1C	0x0	PktDrpSts Packet Dropped Status This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. Dependency: This bit is valid in non-Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.
10	RO	0x0	reserved
9	W1C	0x0	BNAIntr BNA (Buffer Not Available) Interrupt The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done. Dependency: This bit is valid only when Scatter/Gather DMA mode is enabled.
8	W1C	0x0	TxfifoUndrn FIFO Underrun Applies to IN endpoints only. The core generates this interrupt when it detects a transmit FIFO under-run condition for this endpoint. Dependency: This interrupt is valid only when both of the following conditions are true: Parameter OTG_ENDED_TX_FIFO==1, Threshold is enabled, OUT Packet Error (OutPktErr). Applies to OUT endpoints only. This interrupt is asserted when the core detects an overflow or a CRC error for an OUT packet. Dependency: This interrupt is valid only when both of the following conditions are true: Parameter OTG_ENDED_TX_FIFO==1, Thresholding is enabled.
7	W1C	0x0	TxFEmp Transmit FIFO Empty This bit is valid only for IN Endpoints. This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register(GAHBCFG.NPTxFEmpLvl)).

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	W1C	0x0	<p>INEPNakEff IN Endpoint NAK Effective Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. This bit is applicable only when the endpoint is enabled. Back-to-Back SETUP Packets Received (Back2BackSETup) Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.</p>
5	W1C	0x0	<p>INTknEPMis IN Token Received with EP Mismatch Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. Status Phase Received For Control Write (StsPhseRcvd) This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode.</p>
4	W1C	0x0	<p>INTknTxFEmp IN Token Received When TxFIFO is Empty Indicates that an IN token was received when the associated TxFIFO periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. OUT Token Received When Endpoint Disabled (OUTTknEPdis) Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	W1C	0x0	<p>TimeOUT Timeout Condition In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the Timeout interrupt is not asserted.</p> <p>Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. SETUP Phase Done (SetUp). Applies to control OUT endpoints only.</p> <p>Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.</p>
2	W1C	0x0	<p>AHBErr AHB Error Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.</p>
1	W1C	0x0	<p>EPDisbld Endpoint Disabled Interrupt Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	W1C	0x0	XferCompl Transfer Completed Interrupt Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

**USBOTG\_DOEPTSIZn**

Address: Operational Base + offset (0x0b10)

Device endpoint n transfer size register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
30:29	RW	0x0	<p>MC Multi Count Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per micro-frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.</p> <p>2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets</p> <p>For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp). Received Data PID (RxDPID)</p> <p>Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint.</p> <p>2'b00: DATA0 2'b01: DATA2 2'b10: DATA1 2'b11: MDATA</p> <p>SETUP Packet Count (SUPCnt). Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive.</p> <p>2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets</p>
28:19	RW	0x000	<p>PktCnt Packet Count Indicates the total number of USB packets that constitute the Transfer Size amount of data for this endpoint. The power-on value is specified for Width of Packet Counters (parameter OTG_PACKET_COUNT_WIDTH).</p> <p>IN Endpoints: This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO. OUT Endpoints: This field is decremented every time a packet (maximum size or short packet) is written to the RxFIFO.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
18:0	RW	0x00000	XferSize Transfer Size This field contains the transfer size in bytes for the current endpoint. The power-on value is specified for Width of Transfer Size Counters (parameter OTG_TRANS_COUNT_WIDTH). The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. IN Endpoints: The core decrements this field every time a packet from the external memory is written to the TxFIFO. OUT Endpoints: The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.

**USBOTG\_DOEPDMAN**

Address: Operational Base + offset (0x0b14)

Device Endpoint-n DMA Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	DMAAddr DMA Address Holds the start address of the external memory for storing or fetching endpoint data. Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.

**USBOTG\_DOEPDMABn**

Address: Operational Base + offset (0x0b1c)

Device endpoint-n DMA buffer address register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	DMABufferAddr DMA Buffer Address Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

**USBOTG\_DOEPCCTLn**

Address: Operational Base + offset (0x0b20)

Device endpoint-n control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RWSC	0x0	EPEna Endpoint Enable Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled, For IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. For OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. When Scatter/Gather DMA mode is enabled-such as for buffer-pointer based DMA mode: For IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint; For OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. The core clears this bit before setting any of the following interrupts on this endpoint: SETUP Phase Done, Endpoint Disabled, Transfer Completed. Note: For control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
30	R/WSC	0x0	<p>EPDis Endpoint Disable Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint.</p>
29	RO	0x0	<p>SetD1PID Field0001 Abstract Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr). Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro) frame (EO_FrNum) field to odd (micro) frame. This field is not applicable for Scatter/Gather DMA mode.</p>
28	WO	0x0	<p>SetD0PID Set DATA0 PID Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro) frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro) frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receipt descriptor structure.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27	WO	0x0	SNAK Set NAK Applies to IN and OUT endpoints. A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.
26	WO	0x0	CNAK Clear NAK Applies to IN and OUT endpoints. A write to this bit clears the NAK bit for the endpoint.
25:22	RW	0x0	TxFNum TxFIFO Number Shared FIFO Operation: non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO; Others: Specified Periodic TxFIFO number. Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint using coreConsultant, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation: these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
21	RW	0x0	<p>Stall STALL Handshake</p> <p>Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p> <p>Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>
20	RW	0x0	<p>Snp Snoop Mode</p> <p>Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>
19:18	RW	0x0	<p>EPType Endpoint Type</p> <p>Applies to IN and OUT endpoints. This is the transfer type supported by this logical endpoint.</p> <p>2'b00: Control 2'b01: Isochronous 2'b10: Bulk 2'b11: Interrupt</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17	RO	0x0	<p>NAKSts NAK Status Applies to IN and OUT endpoints. Indicates the following:</p> <ul style="list-style-type: none"> <li>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</li> <li>1'b1: The core is transmitting NAK handshakes on this endpoint.</li> </ul> <p>When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.</p>
16	RO	0x0	<p>DPID Endpoint Data PID Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetDOPID fields of this register to program either DATA0 or DATA1 PID.</p> <ul style="list-style-type: none"> <li>1'b0: DATA0</li> <li>1'b1: DATA1</li> </ul> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro) Frame (EO_FrNum). In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro) frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <ul style="list-style-type: none"> <li>1'b0: Even (micro)frame</li> <li>1'b1: Odd (micro)frame</li> </ul> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RWSC	0x0	USBActEP USB Active Endpoint Applies to IN and OUT endpoints. Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.
14:11	RW	0x0	NextEp Next Endpoint Applies to non-periodic IN endpoints only. Indicates the endpoint number to be fetched after the data for the current endpoint is fetched. The core can access this field, even when the Endpoint Enable (EPEna) bit is low. This field is not valid in Slave mode operation. Note: This field is valid only for Shared FIFO operations.
10:0	RW	0x000	MPS Maximum Packet Size Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

**USBOTG\_PCGCR**

Address: Operational Base + offset (0x0b24)

Power and clock gating control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RW	0x0802e	<p>RestoreValue Restore Value (Applicable only when Hibernation is enabled (OTG_EN_PWROPT=2). Defines port clock select for different speeds.</p> <ul style="list-style-type: none"> <li>[31] if_dev_mode           <ul style="list-style-type: none"> <li>- 1: Device mode, core restored as device</li> <li>- 0: Host mode, core restored as host</li> </ul> </li> <li>[30:29] p2hd_prt_spd (PRT speed)           <ul style="list-style-type: none"> <li>- 00: HS</li> <li>- 01: FS</li> <li>- 10: LS</li> <li>- 11: Reserved</li> </ul> </li> <li>[28:27] p2hd_dev_enum_spd (Device enumerated speed)           <ul style="list-style-type: none"> <li>- 00: HS</li> <li>- 01: FS (30/60 MHz clock)</li> <li>- 10: LS</li> <li>- 11: FS (48 MHz clock)</li> </ul> </li> <li>[26:20] mac_dev_addr (MAC device address) Device address</li> <li>[19] mac_termselect (Termination selection)           <ul style="list-style-type: none"> <li>- 0: HS_TERM (Program for High Speed)</li> <li>- 1: FS_TERM (Program for Full Speed)</li> </ul> </li> <li>[18:17] mac_xcvrselect (Transceiver select)           <ul style="list-style-type: none"> <li>- 00: HS_XCVR (High Speed)</li> <li>- 01: FS_XCVR (Full Speed)</li> <li>- 10: LS_XCVR (Low Speed)</li> <li>- 11: LFS_XCVR (Reserved)</li> </ul> </li> <li>[16] sh2pl_prt_ctl[0]           <ul style="list-style-type: none"> <li>- 1: port_power enabled</li> <li>- 0: port_power disabled</li> </ul> </li> <li>[15:14] prt_clk_sel (Refer prt_clk_sel table)</li> </ul>
13	RW	0x0	<p>EssRegRestored Essential Register Values Restored (Applicable only when Hibernation is enabled (OTG_EN_PWROPT=2). When a value of 1 is written to this field, it indicates that register values of essential registers have been restored.</p>
12:10	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RO	0x0	<p>RestoreMode Restore Mode (Applicable only when Hibernation is enabled (OTG_EN_PWR_OPT=2). The application should program this bit to specify the restore mode during RESTORE POINT before programming PCGCCTL.EssRegRest bit is set.</p> <p>Host Mode: 1'b0: Host Initiated Resume, Host Initiated Reset 1'b1: Device Initiated Remote Wake up</p> <p>Device Mode: 1'b0: Device Initiated Remote Wake up 1'b1: Host Initiated Resume, Host Initiated Reset</p>
8	RW	0x0	<p>ResetAfterSusp Reset After Suspend Applicable in Partial power-down mode. In partial power-down mode of operation, this bit needs to be set in host mode before clamp is removed if the host needs to issue reset after suspend. If this bit is not set, then the host issues resume after suspend. This bit is not applicable in device mode and non-partial power-down mode. In Hibernation mode, this bit needs to be set at RESTORE_POINT before PCGCCTL.EssRegRestored is set. In this case, PCGCCTL.restore_mode needs to be set to wait_restore.</p>
7	RO	0x0	<p>L1Suspended Deep Sleep This bit indicates that the PHY is in deep sleep when in L1 state.</p>
6	RO	0x0	<p>PhySleep PHY in Sleep This bit indicates that the PHY is in the Sleep state.</p>
5	RW	0x0	<p>Enbl_L1Gating Enable Sleep Clock Gating When this bit is set, core internal clock gating is enabled in Sleep state if the core cannot assert utmi_l1_suspend_n. When this bit is not set, the PHY clock is not gated in Sleep state.</p>
4	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	RstPdwnModule Reset Power-Down Modules This bit is valid only in Partial Power-Down mode. The application sets this bit when the power is turned off. The application clears this bit after the power is turned on and the PHY clock is up.
2	RW	0x0	PwrClmp Power Clamp This bit is valid only in Partial Power-Down mode (OTG_EN_PWROPT = 1). The application sets this bit before the power is turned off to clamp the signals between the power-on modules and the power-off modules. The application clears the bit to disable the clamping before the power is turned on.
1	RW	0x0	GateHclk Gate Hclk The application sets this bit to gate hclk to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.
0	RW	0x0	StopPclk Stop Pclk The application sets this bit to stop the PHY clock (phy_clk) when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

## 18.7 Interface description

Table 18-2 USB OTG 2.0 Interface Description

<b>Module Pin</b>	<b>Direction</b>	<b>Pad Name</b>	<b>pinmux</b>
OTG_VSSAC	AG	OTG_VSSAC	-
OTG_DVSS	DG	OTG_DVSS	-
OTG_DVDD	DP	OTG_DVDD	-
OTG_VDD25	AP	OTG_VDD25	-
OTG_DM	A	OTG_DM	-

OTG_RKELVIN	A	OTG_RKELVIN	-
OTG_DP	A	OTG_DP	-
OTG_VSSA	AG	OTG_VSSA	-
OTG_VBUS	A	OTG_VBUS	-
OTG_VDD33	AP	OTG_VDD33	-
OTG_ID	A	OTG_ID	-
OTG_drv_vbus	O	IO_PWM2t0_JTA Gtck_OTGdrv_v bus_GPIO4c4	

**Note:** **A**—Analog pad ; **AP**—Analog power; **AG**—Analog ground ; **DP**—Digital power ; **DG**— Digital ground;

## 18.8 Application Note

### 18.8.1 Resume from Suspend Mode

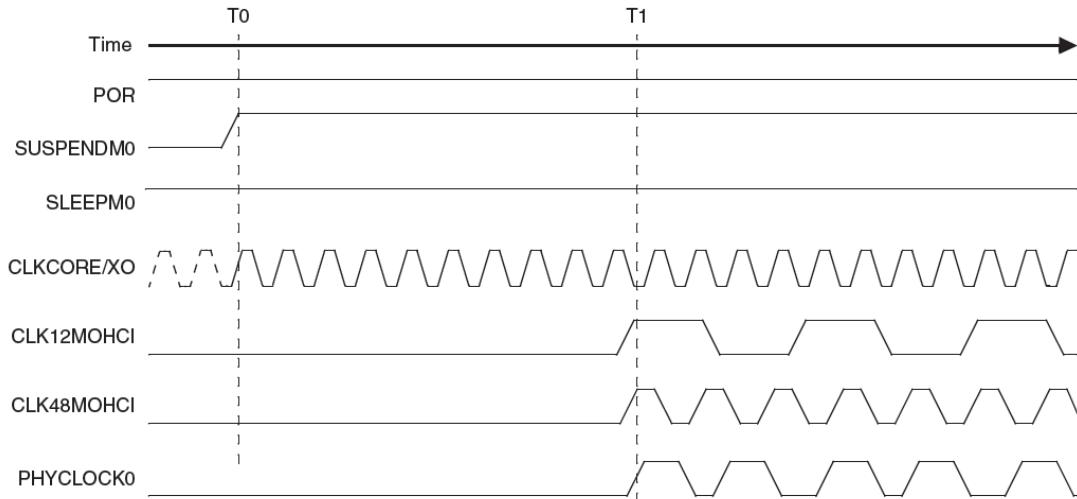


Fig. 18-12 Resume Timing Sequence

When COMMONONNN = 1'b1,  $T_1 < T_0 + 805 \text{ us}$

When COMMONONNN = 1'b0,  $T_1 < T_0 + 16 \text{ us}$

### 18.8.2 Reset a port

Because the assertion of PORTRESET can occur during data reception or transmission, PORTRESET must be de-asserted as follows:

- ❖ Reception:
  - ◆ FS device: After a minimum of 3  $\mu\text{s}$  of stable SE0 on LINESTATE [1:0]
  - ◆ FS/LS host: After a minimum of 8 bit times of J state on LINESTATE [1:0]
  - ◆ HS host/device: A minimum of 150  $\mu\text{s}$  after asserting PORTRESET
- ❖ Transmission:
  - ◆ FS device: After the controller sets both TXVALID and TXVALIDH to 1'b0, followed by a minimum of 3  $\mu\text{s}$  of stable SE0 on LINESTATE [1:0]

- ◆ FS/LS host: After the controller sets both TXVALID0 and TXVALIDH0 to 1'b0, followed by a minimum of 8 bit times of J state on LINESTATE [1:0]
- ◆ HS host/device: A minimum of 150  $\mu$ s after the controller sets both TXVALID0 and TXVALIDH0 to 1'b0. The preceding requirements ensure that there is no activity on the USB when PORTRESET0 is de-asserted.

To avoid any data glitches during port reset, the controller must place the USB 2.0 PHY into a safe state. A safe state for host and device ports is defined as follows:

- ❖ Host: The USB 2.0 PHY is set to Non-Driving (OPMODE [1:0] = 2'b01), and the 15-k $\Omega$  pull-down resistors are enabled (DPPULLDOWN and DMPULLDOWN = 1'b1).
- ❖ Device: The USB 2.0 PHY is set to Non-Driving (OPMODE [1:0] = 2'b01), which disconnects the 1.5-k $\Omega$  resistor from the D+ line.

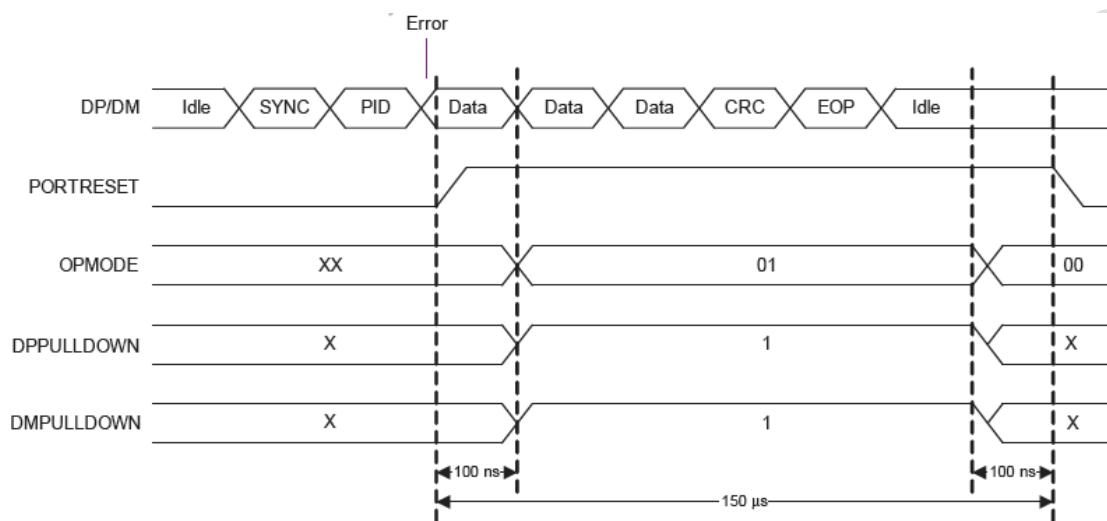


Fig. 18-13 Reset a port when receiving

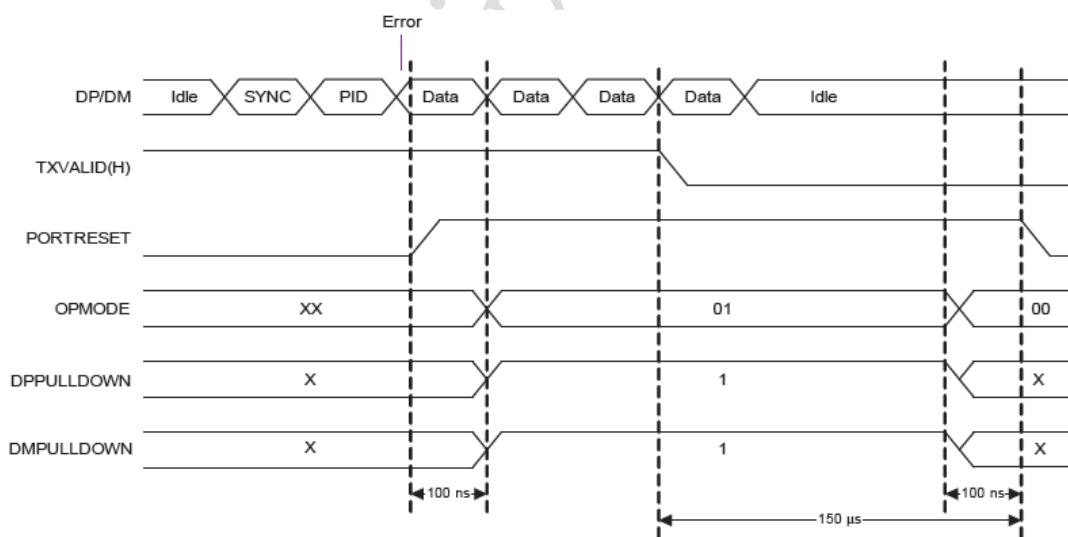


Fig. 18-14 Reset a port when transmitting

### 18.8.3 VBUS threshold

High (min) = 1.7 V  
Low (max) = 0.9 V

## **Chapter 19 USB Host 2.0(0)**

Will update soon!

Rockchip Confidential

## Chapter 20 USB Host 2.0(1)

### 20.1 Overview

USB HOST2.0 supports Non\_OTG Host functions and is fully compliant with USB2.0 specification, and support high-speed (480Mbps), full-speed (12Mbps), low-speed (1.5Mbps) transfer. It is optimized for point-to-point applications (no hub, direct connection to device).

The USB HOST 2.0 supports following features:

- Compliant with the USB2.0 Specification
- Operates in Non\_OTG Host mode
- Operates in High-Speed, Full-Speed, Low-speed mode
- Support 16 channels in host mode
- Built-in one 1024x35 bits FIFO
- Internal DMA with no scatter/gather function
- Support dynamic FIFO adustment

### 20.2 Block Diagram

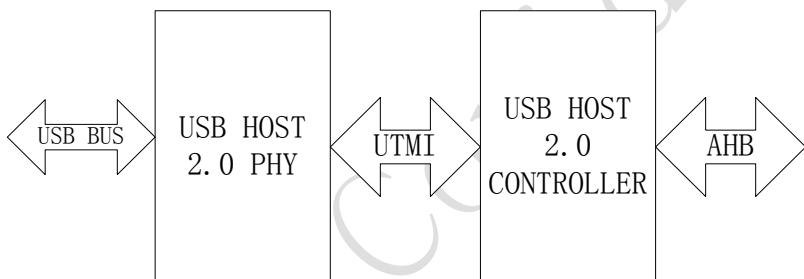


Fig. 20-1 USB HOST 2.0 Architecture

Fig.21-1 shows the architecture of USB HOST 2.0. It is broken up into two separate units: USB HOST 2.0 controller and USB HOST 2.0 PHY. The two units are interconnected with 16-bits UTMI interface.

### 20.3 USB Host2.0 Controller

Much the same as USB OTG with no Device Mode supported. See Chapter 19 for more information.

### 20.4 USB Host2.0 PHY

USB PHY supports dual OTG ports' functions and is fully compliant with USB2.0 specification, and support High-speed (480Mbps), full-speed (12Mbps), low-speed (1.5Mbps) transfer. It provides a complete on-chip transceiver physical solution with ESD protection. A minimum number of external components are needed, which include a 45 ohm resistor for resistance calibration purpose. Its feature contains:

- Fully compliant with USB specifications Rev 2.0, 1.1 HOST V1.2.
- Supports 480Mbps (HS), 12Mbps (FS) & 1.5Mbps(LS) serial data transmission
- Supports low latency hub mode with 40 bit time round trip delay
- 16 bit UTMI interface compliant with UTMI+ specification level 3 Rev 1.
- Loop back BIST mode supported

- Built-in I/O and ESD structure
- On-die self-calibrated HS/FS/LS termination
- 12MHz crystal oscillator with integrated phase-locked loop (PLL) oscillator
- Manufactured in SMIC 65/55nm LL process
- Dual 3.3V / 1.2V supply

### 20.4.1 Block Diagram

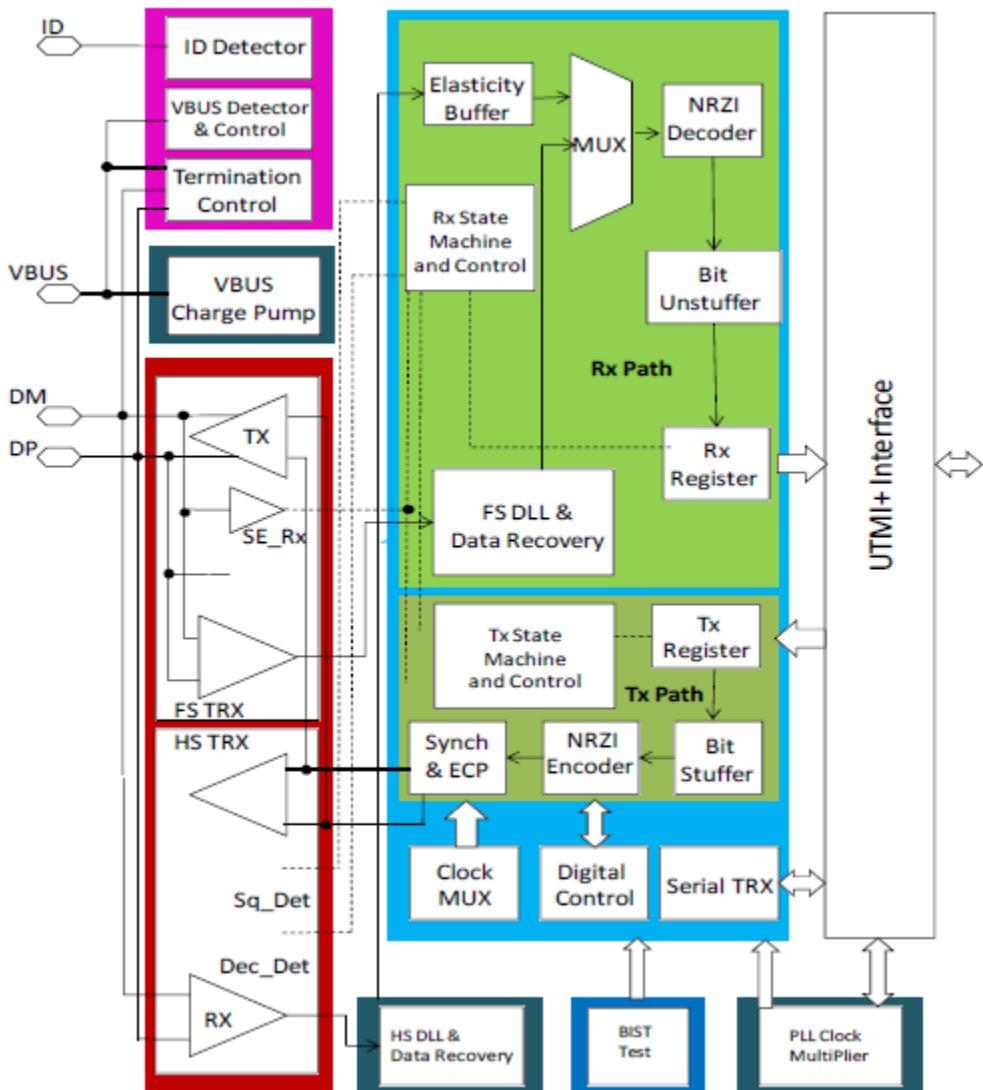


Fig. 20-2 usb phy architecture

#### HS AFE

The HS AFE contains the low-level analog circuitry, and also the HS differential data transmitter and receiver, to perform HS transmission envelope detection and host disconnection detection. It works in HS mode only.

#### HS Transmit driver

The HS transmit driver is active only when transmit is asserted. In HS transceiver enabled mode and the transmit state machine has data to send, the XCVR selects input. Data from transmit data path will be driven onto the DP/DM signal lines when enabled.

#### HS Differential Receiver

When enabled, received HS data will be multiplexed through the receive data path to the receive shift and hold registers. It is active only in HS mode.

#### transmission envelope detector (Squelch detector)

When the amplitude of the differential signal at a receiver's inputs falls below the squelch threshold, the envelope detector will indicate the invalid data. It must indicate squelch when the signal drops below 100mV differential amplitude, and also, it must indicate that the line is not in the squelch state when the signal exceeds 150mV differential amplitude.

#### **Disconnection envelope detector**

In host mode, this envelope detector is active to detect the high speed disconnect state on the line. Disconnection must be indicated when the amplitude of the differential signal at the downstream facing driver's connector is more than 625 mV, and it must not be indicated when the signal amplitude is less than 525 mV.

#### **FS/LS AFE**

In FS or LS mode, the FS/LS AFE is active to send and receive the FS or LS data on the USB bus. Also it supports the reset, suspend and resume detection through the data line single ended receivers.

#### **FS/LS Transmitter**

The FS/LHS transmitter is active only when transmit is asserted. In FS or LS transceiver enabled mode and the transmit state machine has data to send, the XCVR selects input. Data from transmit data path will be driven onto the DP/DM signal lines when enabled.

#### **FS/LS Differential Receiver**

When enabled, received FS or LS data will be multiplexed through the receive data path to the receive shift and hold registers. It is active only in FS or LS modes.

#### **Single ended receivers**

The single ended receivers are used for low-speed and full-speed signaling detection.

#### **Digital Core TX Path**

The digital core TX path has some blocks responsible for SYNC and EOP generation, data encoding, bit stuffing and data serialization. And meanwhile, also a TX state machine is involved to manage the communication with the controller.

#### **TX Shift/Hold Register**

The TX shift/Hold register module consists of an 8-bit primary shift register for parallel/serial conversion and 8-bit hold register used to buffer the next data to serialize. This module is responsible for reading parallel data from the parallel application bus interface upon command and serializing for transmission over USB.

#### **Bit stuffer**

To ensure adequate signal transitions, when sending a packet on USB, a bit stuffer is employed by the transmitter. A '0' has to be inserted after every six consecutive ones in the data stream before the data in NRZI encoded, to force a transition in the NRZI data stream.

**NRZI Encoder**

The High speed, Full speed or low speed serial transmitted data are encoded by The NRZI encoder. As a state transition, a '0' is encoded, and as no state transition, a '1' is encoded.

**Transmit state machine**

The communication between the controller and the PHY in TX path is controlled by the transmit state machine, which synchronizes the Data with the Sync and the EOP, and also supports the LS, FS and HS Modes.

**Digital Core RX Path**

The digital core RX path includes blocks responsible for SYNC and EOP detection and stripping, data decoding, bit un-stuffing and data de-serialization. Also a RX state machine is involved to manage the communication with the controller. FS/LS data and clock is recovered in this section.

**Elasticity buffer**

To compensate for differences between transmitting and receiving clocks, the Elasticity Buffer is used to synchronize the HS extracted data with the PLL internal clock.

**Mux**

The Mux block allows the data from the HS or FS/LS receivers to be routed to the shared receive logic. The state of the Mux is determined by the Xcvr Select input.

**NRZI Decoder**

The NRZI is responsible for decoding the High speed or Full speed received NRZI encoded data. A change in level is decoded as '0' and no change in level is decoded as '1'.

**Bit Un-stuffer**

The Bit Un-stuffer not only recognizes the stuffed bits from the data stream, but also discards them. Also it detects bit stuff error, which is interpreted as HS EOP.

**RX Shift/Hold Register**

This module de-serializes received data and transmits 8-bit parallel data to the application bus interface. It consists of an 8-bit primary shift register for serial to parallel conversion and an 8-bit hold register for buffering the last de-serialized data byte.

**Receiver state machine**

The receiver state machine controls the communication between the controller and the PHY in the RX path, strips the SYNC and the EOP from the Data and supports the LS, FS and HS Modes.

**PLL Clock Multiplier**

This module is composed of the off-chip crystal and the on-chip clock multiplier. It generates the appropriate internal clocks for the UTM and the CLK output signal. All data transfer signals are synchronized with the CLK signal.

**External Crystal**

The external crystal is composed of a precise resonance frequency crystal and a crystal oscillator. It is optional to have this crystal oscillator integrated on-chip or have it off-chip. This crystal/crystal oscillator provides a very precise clock of 12 MHz with deviation of  $\pm 100$  ppm. The oscillator is not a part of the PHY, but external.

### **Clock Multiplier**

The UTM interface is described as an un-directional/bi-directional 8-bit/16-bit parallel interface and the CLK signal is a 60/30 MHz signal. All data transfer signals should be synchronized with the CLK signal. CLK usable signal is internally implemented which blocks any transitions of CLK until it is usable. Meanwhile, the clock multiplier provides another three clocks in addition to the CLK signal. That is a 480 MHz and 7.5 MHz clock signals.

### **Clock MUX**

The Clock Multiplexer supplies both the transmitter and receiver paths with the adequate bit clock depending on the XcvrSelect signal and to ensure smooth clock switching. It also includes clock gating and power-down features.

### **Control Logic Block**

This block is responsible for controlling, enabling and disabling the different blocks in the system.

### **OTG Circuitry (optional)**

With the OTG circuitry, the system has the capability to dynamically switch between host and peripheral, enable dual role device behavior and point-to-point communication. The OTG circuitry functions as VBUS generation and detection. Both ID detection and terminations control are implemented in it.

### **ID Detector (optional)**

To provide the ID signal that is used to indicate the state of the ID pin on the USB mini receptacle. This pin makes it able to determine which kind of plug is connected and to confirm if the device default state is A device or B device.

### **VBUS Detector and termination control**

The VBUS detector is a set of comparators, functions to monitor and sense the voltage on USB bus power line. For VBUS signaling and discharging, VBUS pull up and pull-down resistors are also implemented.

### **Automatic Test Functions**

- Loop-back test to address all IP components.

In loop-back test mode, all transmitted data packets are received back in an internal loop to check IP functional integrity. There are some digital components that cannot be tested with the scan technique due to the high-speed nature of the digital part. To be regarded as a good idea, Loop-back allows testing full design paths at speed. It should complement the testing suite for digital core to achieve the highest coverage possible. According to the UTMI specification Section 5.18, version 1.05 Page 34, the 8 bits un-directional data bus can be implemented as 8 bits bi-directional one. This implementation will hinder the loop-back test functionality.

USB Host2.0 PHY doesn't support UART-DEBUG function.

## 20.5 Register Description

The Controller Registers are much the same as OTG with no Device-Mode supported. The registers of Device are not available. See Ch26 for more information.

The phy registers are in GRF. Please refer to this chapter for more detail.

## 20.6 Interface description

Table 20-1 USB HOST 2.0 Interface Description

Module Pin	Direction	Pad Name	pinmux
VSSA	AG	IO_INNO_USB_VSSA	-
VCCA3P3	AP	IO_INNO_USB_VCCA3P3	-
VCCCORE1P0	AP	IO_INNO_USB_VCCCORE1P0	-
VCC18	AP	IO_INNO_USB_VCC18	
USB0PN	A	IO_INNO_USB_PN	-
USBRBIAS	A	IO_INNO_USB_RBIAS	-
USB0PP	A	IO_INNO_USB_PP	-
VBUS	A	IO_INNO_USB_VBUS	-
USB0ID	A	IO_INNO_USB_ID	-

**Note:** **A**—Analog pad ; **AP**—Analog power; **AG**—Analog ground ;**DP**—Digital power ;**DG**—Digital ground;

## 20.7 Application Note

See Ch OTG for more information.

## Chapter 21 USB HSIC

### 21.1 Overview

USB HSIC Controller is fully compliant with the High-Speed Inter-Chip USB Electrical Specification, and supports highspeed (480-Mbps) transfers.

USB HSIC supports following features:

- Compliant with the High-Speed Inter-Chip USB Electrical Specification and Enhanced Host Controller Interface Specification 2.0
- 1 Port HSIC PHY Interface Operates in host mode
- Built-in one 512x64 bits data FIFO and one 68x32 bits descriptor FIFO
- Internal DMA with scatter/gather function

### 21.2 Block Diagram

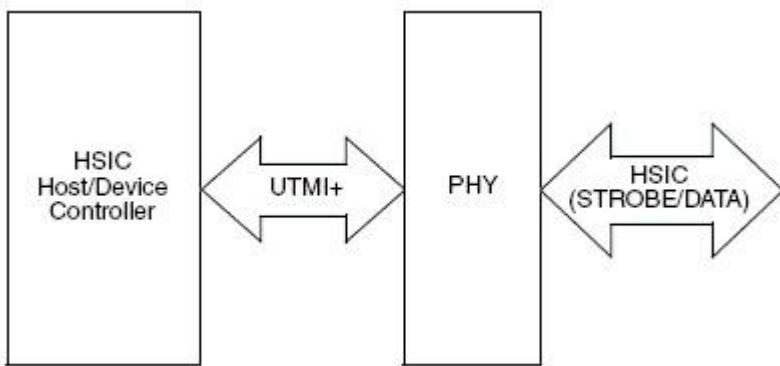


Fig. 21-1 USB HSIC Architecture

Fig. 22-1 is the architecture of USB HSIC. It is broken up into two separate units: USB HSIC Controller and USB HSIC PHY. The two units are interconnected with UTMI interface.

### 21.3 USB HSIC Controller

USB HSIC Controller is an USB host controller, which supports only high-speed (480Mbps) using an EHCI Host Controller and is fully compliant with USB2.0specification. This controller will support UTMI+ Level 3 PHY interface. It connects to the industry-standard AMBA AHB for communication with the application and system memory. And it is optimized for portable electronic device and point-to-point applications (no hub, direct connection to device).

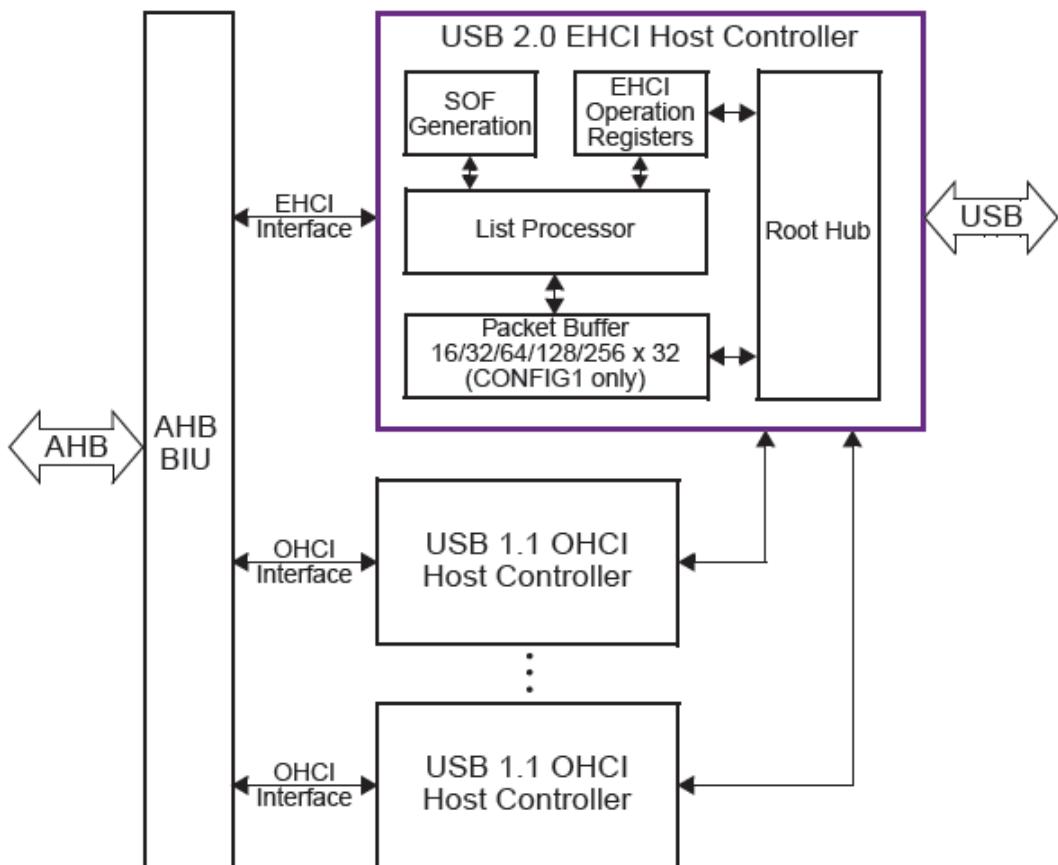


Fig. 21-2 USB HSIC Controller Architecture

**AHB BIU :****AHB Slave Bus Interface Unit**

The controller supports an AHB interface to the application. The BIU is divided into Master and Slave modules.

OHCI is not supported.

The Slave BIU provides the Enhanced Host Controller Driver (EHCD) interface to write to and read from the Operational registers through the AHB. For EHCI register access, two AHB clock cycle wait states are introduced. The EHCI states do not change the latency. The Slave BIU is designed to target only Operational register accesses, which do not support burst accesses, or retry and split responses.

**AHB Master Bus Interface Unit (BIU)**

The Master BIU block receives requests from the List Processor and reads data from or writes data to system memory through the AHB. This block fetches descriptors, data, and status.

**List Processor**

The List Processor block is the main controller. This block is implemented with multiple state machines to perform the list service flow, which is set up by the Host Controller Driver according to the priority set in the Operational registers. In addition, this block is implemented with a controller that interfaces with the BIU, Packet Buffer, EHCI Operational registers, SOF Generators, and the Root Hub. The List Processor has the following sub modules:

- Master Controller Unit (LPMCU)
- List Management Unit (LPLMU)
- List Service MUX (LPLSM)

- iTD Servicer (LPITD)
- qTD and QH Servicer (LPQTD)
- SiTD Servicer (LPSITD)
- System Memory Controller (LPSMC)
- Data Structure Cache (LPDSC)

## **Operational Registers**

The EHCI Capability and Operational registers are stored in this block. The registers in the Auxiliary Well are part of the Operational registers, but are implemented in a separate module.

### **Start-of-Frame (SOF) Generator**

SOF packets are generated with an SOF counter to generate micro-SOFs for each micro-frame. Micro-frame duration is derived from the Frame Length Adjustment (FLADJ) register value, which can be configured through the Application Strap Signals interface. The same FLADJ values must be configured through strap signals. This ensures that the host micro-frame duration and per-port micro-frame duration remain the same.

### **Packet Buffer (PBUF)**

The Packet Buffer provides storage and control for IN/OUT data. During an OUT transaction, the LPSMC fetches data from the system memory and writes it here. During an IN transaction, the data is written by the Root Hub.

### **Root Hub (RH)**

The Root Hub propagates Reset and Resume signals to downstream ports and handles port connections and disconnections. In addition, the Root Hub is implemented with Port Router logic to route the ports to the EHCI Host Controller. The Root Hub operates on the local PHY clock, which operates on a free-running 30-MHz clock, and the clock source from each physical port, which operate at 30 MHz with a 16-bit interface. The Root Hub has the following sub modules:

- Host Parallel Interface Engine (RHPIE)
- Port Router (RHPRTTR)
- UTMI Block (RHUTM)
- Port State Machine and Port Interface Block (RHPRT)
- Synchronizer Block (RHSYNC)

## **21.4 USB HSIC PHY**

The High-Speed Inter-Chip Universal Serial Bus (USB) 2.0 physical layer connects a USB host controller to an HSIC USB system. The HSIC PHY is a complete mixed-signal IP solution designed to implement USB 2.0 HSIC (High-Speed Inter-Chip) connectivity in a System-on-Chip (SoC) design targeted to a specific fabrication process using core and 1.8-V thick-oxide devices. The HSIC PHY supports the High-Speed Inter-Chip (HSIC) USB protocol and data rate.

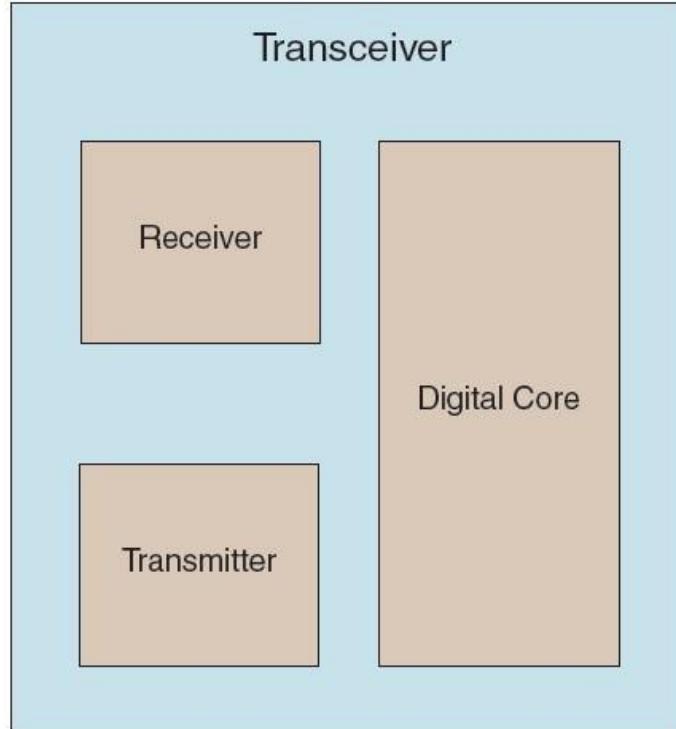


Fig. 21-3 USB HSIC PHY Architecture

Fig.22-3 shows the architecture of USB HSIC PHY. The block comprises the Analog block and a digital core, which control transaction and receive operations. The Transceiver requires a 480-MHz input clock for data transmission and data recovery.

**Receiver Block** This block contains the HSIC receiver.

**Transmitter Block** The Transmitter block contains the HSIC driver. Top-level parameter override bits are provided to trim the pull-up and pull-down resistances.

Fig.22-4 shows the USB HSIC PHY Digital block diagram for a one-port macro.

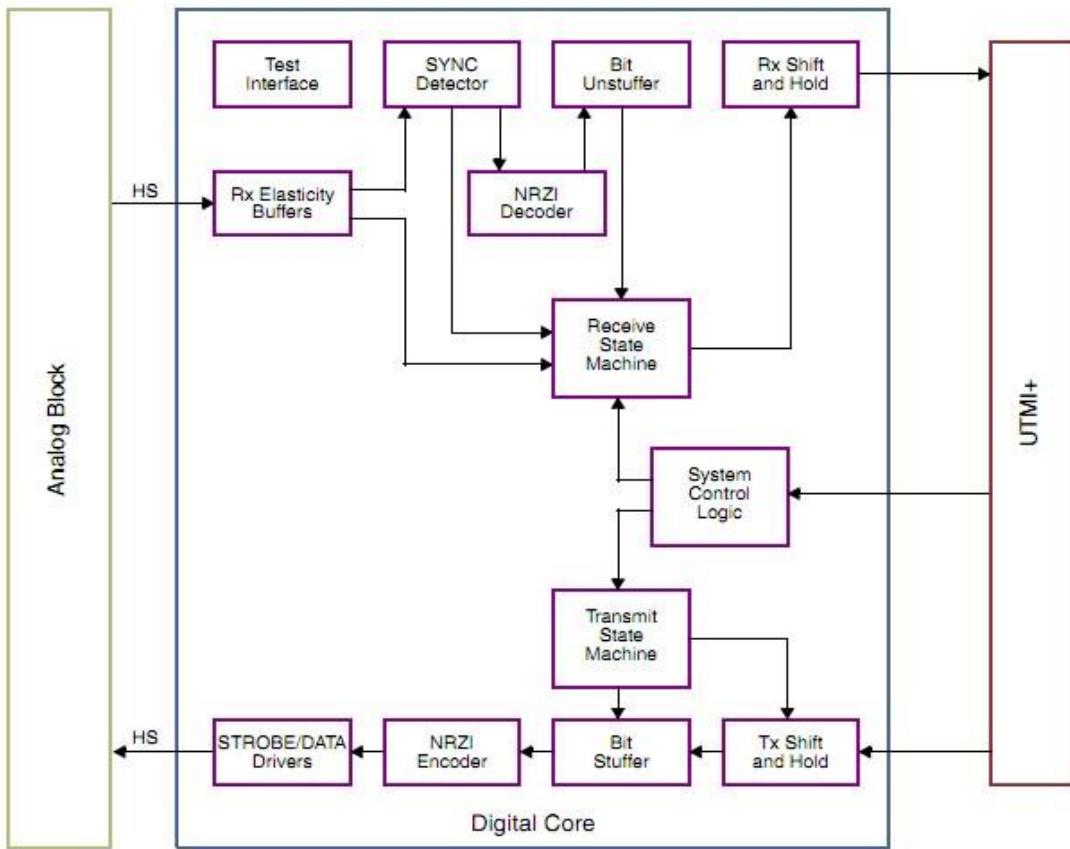


Fig. 21-4 USB HSIC PHY Digital Block Architecture

**SYNC Detector** The SYNC Detector detects 12-32 bits of SYNC data. After detecting the pattern, the detector recognizes the start of the Packet ID (PID) field. The packet data content after the SYNC field is passed to other blocks.

**NRZI Decoder:** The NRZI Decoder decodes NRZI-encoded packet data.

**Bit Un-stuffer:** The Bit Un-stuffer removes 0-stuffed bits from received, data packets as per the HSIC specification. The Bit Un-stuffer also detects bit-stuffing errors in the data packet.

**Receive Shift/Hold Registers:** The Receive Shift/Hold registers convert serial, received data from the Bit Un-stuffer to a parallel data stream and transmits 8/16-bit parallel data to the UTMI+ parallel receive port. This module consists of an 8/16-bit primary shift register for serial-to-parallel conversion and an 8/16-bit hold register to buffer the last de-serialized data byte.

**Receive State Machine:** The inputs to the Receive state machine are from the SYNC Detector, RX Elasticity Buffers, and Bit Unstuffer. The receive state machine generates signals for the parallel receive port that indicates the validity of parallel data.

**System Control Logic:** The System Control Logic accepts control inputs from the UTMI+. The control inputs configure the speed of operation and the HSIC PHY's termination resistors.

**Transmit State Machine:** The Transmit state machine handles handshake signals at the UTMI+ parallel transmit port in accordance with the UTMI+ specification. Transmit Shift/Hold Registers This module is responsible for reading parallel data from the UTMI+ parallel transmit port and serializing the data for transmission over the HSIC USB. This module consists of an 8/16-bit primary shift register for parallel-to-serial conversion and an 8/16-bit hold register to buffer subsequent data to be serialized.

**Bit Stuffer:** The Bit Stuffer inserts 0 data-bits into the transmit data stream as per the HSIC specification. The inserted 0 forces a transition in the NRZI data stream. The transition is necessary for data recovery. Bit stuffing is enabled, starting with the SYNC pattern, and applied throughout the transmission. The 1 data bit that ends the SYNC pattern is counted as the first data bit in the sequence. HS EOP transmissions are not bit-stuffed.

**NRZI Encoder:** This block NRZI-encodes the serial data to be transmitted. NRZI encoding forces a level transition whenever a 0 data bit is input; a 1 data bit results in the previous level being maintained.

## 21.5 Register description

### 21.5.1 Register Summary

Name	Offset	Size	Reset Value	Description
HSIC_HCCAPBASE	0x0000	W	0x01000010	
HSIC_HCSPARAMS	0x0004	W	0x00001111	Structural Parameters
HSIC_HCCPARAMS	0x0008	W	0x0000a026	Capability Parameters
HSIC_USBCMD	0x0010	W	0x00000000	USB Command Register
HSIC_USBSTS	0x0014	W	0x00001000	USB Status Register
HSIC_USBINTR	0x0018	W	0x00000000	USB Interrupt Enable Register
HSIC_FRINDEX	0x001c	W	0x00000000	Frame Index Register
HSIC_CTRLDSSEGMENT	0x0020	W	0x00000000	Control Data Structure Segment Register
HSIC_PERIODICLISTBASE	0x0024	W	0x00000000	Periodic Frame List Base Address Register
HSIC_ASYNCLISTADDR	0x0028	W	0x00000000	Current Asynchronous List Address Register
HSIC_CONFIGFLAG	0x0050	W	0x00000000	Configure Flag Register
HSIC_PORTSC_1	0x0054	W	0x00000000	Port Status and Control Register 1
HSIC_INSNREG00	0x0090	W	0x00000000	Programmable Micro-frame Base Value
HSIC_INSNREG01	0x0094	W	0x00000000	Programmable Packet Buffer OUT/IN Thresholds
HSIC_INSNREG02	0x0098	W	0x00000000	Programmable Packet Buffer Depth
HSIC_INSNREG03	0x009c	W	0x00000000	Register0000 Abstract
HSIC_INSNREG04	0x00a0	W	0x00000000	Register0000 Abstract
HSIC_INSNREG05	0x00a4	W	0x00000000	Control and Status Register
HSIC_INSNREG06	0x00a8	W	0x00000000	AHB Error Status
HSIC_INSNREG07	0x00ac	W	0x00000000	AHB Master Error Address
HSIC_INSNREG08	0x00b0	W	0x00000000	HSIC Enable/Disable

Notes: Size: **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

## 21.5.2 Detail Register Description

### HSIC\_HCCAPBASE

Address: Operational Base + offset (0x0000)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0100	HCIVERSION Host Controller Interface Version Number This is a two-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision
15:8	RO	0x0	reserved
7:0	RO	0x10	CAPLENGTH Capability Registers Length This register is used as an offset to add to register base to find the beginning of the Operational Register Space.

### HSIC\_HCSPARAMS

Address: Operational Base + offset (0x0004)

Structural Parameters

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23:20	RO	0x0	Debug_Port_Num Debug Port Number Optional. This register identifies which of the host controller ports is the debug port. The value is the port number (one-based) of the debug port. A nonzero value in this field indicates the presence of a debug port. The value in this register must not be greater than N_PORTS
19:17	RO	0x0	reserved
16	RO	0x0	P_INDICATOR Port Indicators This bit indicates whether the ports support port indicator control. When this bit is a one, the port status and control registers include a read/writeable field for controlling the state of the port indicator

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:12	RO	0x1	<p>N_CC Number of Companion Controller This field indicates the number of companion controllers associated with this USB 2.0 host controller. A zero in this field indicates there are no companion host controllers.</p> <p>Port-ownership hand-off is not supported. Only high-speed devices are supported on the host controller root ports. A value larger than zero in this field indicates there are companion USB 1.1 host controller(s). Port-ownership hand-offs are supported. High, Full- and Low-speed devices are supported on the host controller root ports.</p>
11:8	RO	0x1	<p>N_PCC Number of Ports per Companion Controller This field indicates the number of ports supported per companion host controller. It is used to indicate the port routing configuration to 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.</p>
7	RO	0x0	<p>Rout_Rule Port Routing Rules This field indicates the method used by this implementation for how all ports are mapped to companion controllers. The value of this field has the following interpretation: 1'b0: The first N_PCC ports are routed to the lowest numbered function companion host controller, the next N_PCC port are routed to the next lowest function companion controller, and so on. 1'b1: The port routing is explicitly enumerated by the first N_PORTS elements of the HCSP-PORTROUTE array.</p>
6:5	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RO	0x1	<p>PPC Port Power Control</p> <p>This field indicates whether the host controller implementation includes port power control. A one in this bit indicates the ports have port power switches. A zero in this bit indicates the port do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register</p>
3:0	RO	0x1	<p>N_PORTS</p> <p>This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register Space. Valid values are in the range of 1H to FH. A zero in this field is undefined.</p>

**HSIC\_HCCPARAMS**

Address: Operational Base + offset (0x0008)

Capability Parameters

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:8	RO	0xa0	<p>EECP</p> <p>EHCI Extended Capabilities Pointer</p> <p>Default = Implementation Dependent. This optional field indicates the existence of a capabilities list. A value of 00h indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 40h or greater if implemented to maintain the consistency of the PCI header defined for this class of device.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:4	RO	0x2	Isoc_Schedule_Threshold Isochronous Scheduling Threshold Default = implementation dependent. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data.
3	RO	0x0	reserved
2	RO	0x1	Ayn_Schedule_Park Asynchronous Schedule Park Capability Default = Implementation dependent. If this bit is set to a one, 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.
1	RO	0x1	Prog_Frame_List_Flag Programmable Frame List Flag Default = Implementation dependent. If this bit is set to a zero, then 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 should be set to zero. If set to a one, then 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.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RO	0x0	<p>Addressing_Cap 64-bit Addressing Capability This field documents the addressing range capability of this implementation. The value of this field determines whether software should use the data structures (32-bit) or those (64-bit). Values for this field have the following interpretation:</p> <p>1'b0: data structures using 32-bit address memory pointers 1'b1: data structures using 64-bit address memory pointers</p>

**HSIC\_USBCMD**

Address: Operational Base + offset (0x0010)

USB Command Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23:16	RW	0x00	<p>Int_Threshold_Ctrl Interrupt Threshold Control Default 08h. This field is used by system software to select the maximum rate at which the host controller will issue interrupts. The only valid values are defined below. If software writes an invalid value to this register, the results are undefined. Value Maximum Interrupt Interval 8'h00: Reserved 8'h01: 1 micro-frame 8'h02: 2 micro-frames 8'h04: 4 micro-frames 8'h08: 8 micro-frames (default, equates to 1 ms) 8'h10: 16 micro-frames (2 ms) 8'h20: 32 micro-frames (4 ms) 8'h40: 64 micro-frames (8 ms) Any other value in this register yields undefined results. Software modifications to this bit while HCHalted bit is equal to zero results in undefined behavior.</p>
15:12	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x0	Asyn_Schedule_Park_Ena Asynchronous Schedule Park Mode Enable If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this bit defaults to a 1h and is R/W. Otherwise the bit must be a zero and is RO. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled.
10	RO	0x0	reserved
9:8	RW	0x0	Async_Schedule_Part_Count Asynchronous Schedule Park Mode Count If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. 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. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior.
7	RW	0x0	L_Reset Light Host Controller Reset This control bit is not required. If implemented, it allows the driver to reset the EHCI controller without affecting the state of the ports or the relationship to the companion host controllers. For example, the PORSTC registers should not be reset to their default values and the CF bit setting should not go to zero (retaining port ownership relationships). A host software read of this bit as zero indicates the Light Host Controller Reset has completed and it is safe for host software to re-initialize the host controller. A host software read of this bit as a one indicates the Light Host Controller Reset has not yet completed. If not implemented a read of this field will always return a zero.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RW	0x0	<p><b>Int_Asyn_Doorbell</b>  <b>Interrupt on Async Advance Doorbell</b>  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 state, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Async Advance Enable bit in the USBINTR register is a one then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to a zero after it has set the Interrupt on Async Advance status bit in the USBSTS register to a one. Software should not write a one to this bit when the asynchronous schedule is disabled. Doing so will yield undefined results.</p>
5	RW	0x0	<p><b>Asyn_Shcedule_Ena</b>  <b>Asynchronous Schedule Enable</b>  This bit controls whether the host controller skips processing the Asynchronous Schedule. Values mean:  1'b0 Do not process the Asynchronous Schedule  1'b1 Use the ASYNCLISTADDR register to access the Asynchronous Schedule.</p>
4	RW	0x0	<p><b>Period_Schedule_Ena</b>  <b>Periodic_Schedule_Enable</b>  Default 0b. This bit controls whether the host controller skips processing the Periodic Schedule. Values mean:  1'b0: Do not process the Periodic Schedule  1'b1: Use the PERIODICLISTBASE register to access the Periodic Schedule.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:2	RW	0x0	<p>Frame_List_Size</p> <p>Default 00b. This field is R/W only if Programmable Frame List Flag in the HCCPARAMS registers is set to a one. This field specifies the size of the frame list. The size the frame list controls which bits in the Frame Index Register should be used for the Frame List Current index. Values mean:</p> <ul style="list-style-type: none"> <li>2'b00: 1024 elements (4096 bytes) Default value</li> <li>2'b01: 512 elements (2048 bytes)</li> <li>2'b10: 256 elements (1024 bytes) –for resource-constrained environments</li> <li>2'b11: Reserved</li> </ul>
1	RW	0x0	<p>HCRESET</p> <p>Host Controller Reset</p> <p>This control bit is used by software to reset the host controller. The effects of this on Root Hub registers are similar to a Chip Hardware Reset. When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines, etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. PCI Configuration registers are not affected by this reset. All operational registers, including port registers and port state machines are set to their initial values. Port ownership reverts to the companion host controller(s), with the side effects. Software must reinitialize the host controller in order to return the host controller to an operational state. This bit is set to zero by the Host Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	<p>RS Run/Stop Default 0b. 1=Run. 0=Stop. When set to a 1, the Host Controller proceeds with 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 and any actively pipelined transactions on the USB and then halts. The Host Controller must halt within 16 micro-frames after software clears the Run bit. The HC Halted bit in the status register indicates when the Host Controller has finished its pending pipelined transactions and has entered the stopped state. Software must not write a one to this field unless the host controller is in the Halted state (i.e. HCHalted in the USBSTS register is a one). Doing so will yield undefined results.</p>

**HSIC\_USBSTS**

Address: Operational Base + offset (0x0014)

USB Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15	RO	0x0	<p>Asyn_Schedule_Status Asynchronous Schedule Status 0=Default. The bit reports the current real status of the Asynchronous Schedule. If this bit is a zero then the status of the Asynchronous Schedule is disabled. If this bit is a one then the status of the Asynchronous Schedule 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>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14	RO	0x0	Periodic_Schedule_Status 0=Default. The bit reports the current real status of the Periodic Schedule. If this bit is a zero then the status of the Periodic Schedule is disabled. If this bit is a one then the status of the Periodic Schedule 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).
13	RO	0x0	Reclamation This is a read-only status bit, which is used to detect an empty asynchronous schedule.
12	RO	0x1	HCHalted 1=Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing as a result of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error).
11:6	RO	0x0	reserved
5	W1C	0x0	Int_Asyn_Advance Interrupt on Async Advance 0=Default. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Asynchronous Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source.
4	W1C	0x0	Host_sys_err Host System Error The Host Controller sets this bit to 1 when a serious error occurs during a host system access involving the Host Controller module. In a PCI system, conditions that set this bit to 1 include PCI Parity error, PCI Master Abort, and PCI Target Abort. When this error occurs, the Host Controller clears the Run/Stop bit in the Command register to prevent further execution of the scheduled TDs.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	W1C	0x0	<p>Frame_List_Rollover Frame List Rollover The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example, if the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX[13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FRINDEX[12] toggles.</p>
2	W1C	0x0	<p>Port_Chng Port Change Detect The Host Controller sets this bit to a one when any port for which the Port Owner bit is set to zero has a change bit transition from a zero to a one or a Force Port Resume bit transition from a zero to a one as a result of a J-K transition detected on a suspended port. This bit will also be set as a result of the Connect Status Change being set to a one after system software has relinquished ownership of a connected port by writing a one to a port's Port Owner bit. This bit is allowed to be maintained in the Auxiliary power well. Alternatively, it is also acceptable that on a D3 to D0 transition of the EHCI HC device, this bit is loaded with the OR of all of the PORTSC change bits (including: Force port resume, over-current change, enable/disable change and connect status change).</p>
1	W1C	0x0	<p>USBERRINT USB Error Interrupt The Host Controller sets this bit to 1when completion of a USB transaction results in an error condition (e.g., error counter underflow). If the TD on which the error interrupt occurred also had its IOC bit set, both this bit and USBINT bit are set.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	W1C	0x0	<p>USBINT USB Interrupt</p> <p>The Host Controller sets this bit to 1 on the completion of a USB transaction, which results in the retirement of a Transfer Descriptor that had its IOC bit set.</p> <p>The Host Controller also sets this bit to 1 when a short packet is detected (actual number of bytes received was less than the expected number of bytes).</p>

**HSIC\_USBINTR**

Address: Operational Base + offset (0x0018)

USB Interrupt Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x0	<p>Int_Asyn_Advance_Ena Interrupt on Async Advance Enable</p> <p>When this bit is a one, and the Interrupt on Async Advance bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit.</p>
4	RW	0x0	<p>Host_Sys_Error_Ena Host System Error Enable</p> <p>When this bit is a one, and the Host System Error Status bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Host System Error bit.</p>
3	RW	0x0	<p>Frame_List_Roll_Ena Frame List Rollover Enable</p> <p>When this bit is a one, and the Frame List Rollover bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Frame List Rollover bit.</p>
2	RW	0x0	<p>Port_Chng_Int_Ena Port Change Interrupt Enable.</p> <p>When this bit is a one, and the Port Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port Change Detect bit.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	USB_ERR_INT_ENA USB Error Interrupt Enable When this bit is a one, and the USBERRINT bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit.
0	RW	0x0	USB_INT_ENA USB Interrupt Enable When this bit is a one, and the USBINT bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit.

**HSIC\_FRINDEX**

Address: Operational Base + offset (0x001c)

Frame Index Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>												
31:14	RO	0x0	reserved												
13:0	RW	0x0000	Frame_Index The value in this register increases at the end of each time frame (e.g. micro-frame). Bits [N:3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register. USBCMD [Frame List Size] Elements <table> <tr> <td>00b</td> <td>(1024)</td> <td>12</td> </tr> <tr> <td>01b</td> <td>(512)</td> <td>11</td> </tr> <tr> <td>10b</td> <td>(256)</td> <td>10</td> </tr> <tr> <td>11b</td> <td>Reserved</td> <td></td> </tr> </table>	00b	(1024)	12	01b	(512)	11	10b	(256)	10	11b	Reserved	
00b	(1024)	12													
01b	(512)	11													
10b	(256)	10													
11b	Reserved														

**HSIC\_CTRLDSSEGMENT**

Address: Operational Base + offset (0x0020)

Control Data Structure Segment Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	CTRLDSSEGMENT This 32-bit register corresponds to the most significant address bits [63:32] for all EHCI data structures. If the 64-bit Addressing Capability field in HCCPARAMS is a zero, then this register is not used. Software cannot write to it and a read from this register will return zeros. If the 64-bit Addressing Capability field in HCCPARAMS is a one, then this register is used with the link pointers to construct 64-bit addresses to EHCI control data structures. This register is concatenated with the link pointer from either PERIODICLISTBASE, ASYNCLISTADDR, or any control data structure link field to construct a 64-bit address. This register allows the host software to locate all control data structures within the same 4 Gigabyte memory segment.

**HSIC\_PERIODICLISTBASE**

Address: Operational Base + offset (0x0024)

Periodic Frame List Base Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RW	0x000000	Low Base Address These bits correspond to memory address signals [31:12], respectively
11:0	RO	0x0	reserved

**HSIC\_ASYNCLISTADDR**

Address: Operational Base + offset (0x0028)

Current Asynchronous List Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RW	0x00000000	LPL Link Pointer Low These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH)
4:0	RO	0x0	reserved

**HSIC\_CONFIGFLAG**

Address: Operational Base + offset (0x0050)

Configure Flag Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	<p>CF Configure Flag Default 0b. Host software sets this bit as the last action in its process of configuring the Host Controller. This bit controls the default port-routing control logic. Bit values and side-effects are listed below.</p> <p>1'b0: Port routing control logic default-routes each port to an implementation dependent classic host controller.</p> <p>1'b1: Port routing control logic default-routes all ports to this host controller.</p>

**HSIC\_PORTSC\_1**

Address: Operational Base + offset (0x0054)

Port Status and Control Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:23	RO	0x0	reserved
22	RW	0x0	<p>WKOC_E Wake on Over-current Enable Default = 0b. Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power is zero.</p>
21	RW	0x0	<p>WKDSCNNT_E Wake on Disconnect Enable Default = 0b. Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power is zero.</p>
20	RW	0x0	<p>WKCNNT_E Wake on Connect Enable Default = 0b. Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power is zero.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19:16	RW	0x0	<p>Port_Test_Control</p> <p>Default = 0000b. When this field is zero, the port is NOT operating in a test mode. A non-zero value indicates that it is operating in test mode and the specific test mode is indicated by the specific value. The encoding of the test mode bits are (4'b0110 - 4'b1111 are reserved):</p> <p>Bits Test Mode</p> <p>4'b0000: Test mode not enabled          4'b0001: Test J_STATE          4'b0010: Test K_STATE          4'b0011: Test SE0_NAK          4'b0100: Test Packet          4'b0101: Test FORCE_ENABLE</p> <p>Refer to Section 4.14 for the operational model for using these test modes and the USB Specification Revision 2.0, Chapter 7 for details on each test mode.</p>
15:14	RW	0x0	<p>Port_Indicator_Control</p> <p>Default = 00b. Writing to these bits has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. If P_INDICATOR bit is a one, then the bit encodings are:</p> <p>Bit Value Meaning</p> <p>2'b00: Port indicators are off          2'b01: Amber          2'b10: Green          2'b11: Undefined</p> <p>Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used. This field is zero if Port Power is zero.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13	RW	0x0	<p>Port_Owner</p> <p>Field0000 Abstract</p> <p>This bit unconditionally goes to a 0b when the Configured bit in the CONFIGFLAG register makes a 0b to 1b transition. This bit unconditionally goes to 1b whenever the Configured bit is zero. System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that a companion host controller owns and controls the port.</p>
12	RW	0x0	<p>PP</p> <p>Port Power</p> <p>The function of this bit depends on the value of the Port Power Control (PPC) field in the HCSPARAMS register. The behavior is as follows:</p> <p>PPC PP Operation</p> <p>1'b0: 1b RO. Host controller does not have port power control switches. Each port is hard-wired to power.</p> <p>1'b1: 1b/0b R/W. Host controller has port power control switches. This bit represents the current setting of the switch (0 = off, 1 = on). When power is not available on a port (i.e. PP equals a 0), the port is nonfunctional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller from a 1 to 0 (removing power from the port).</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:10	RO	0x0	<p>Line_Status</p> <p>These bits reflect the current logical levels of the D+ (bit 11) and D-(bit 10) signal lines.</p> <p>These bits are used for detection of low-speed USB devices prior to the port reset and enable sequence. This field is valid only when the port enable bit is zero and the current connect status bit is set to a one. The encoding of the bits are:</p> <p>Bits[11:10] USB State Interpretation</p> <p>2'b00: SE0 Not Low-speed device, perform EHCI reset</p> <p>2'b10: J-state Not Low-speed device, perform EHCI reset</p> <p>2'b01: K-state Low-speed device, release ownership of port</p> <p>2'b11: Undefined Not Low-speed device, perform EHCI reset.</p> <p>This value of this field is undefined if Port Power is zero.</p>
9	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	RW	0x0	<p>Port_Reset</p> <p>1'b1: Port is in Reset 1'b0: Port is not in Reset</p> <p>Default 0. When software writes a one to this bit (from a zero), the bus reset sequence as defined in the USB Specification Revision 2.0 is started. Software writes a zero to this bit to terminate the bus reset sequence. Software must keep this bit at a one long enough to ensure the reset sequence, as specified in the USB Specification Revision 2.0, completes.</p> <p>Note: when software writes this bit to a one, it must also write a zero to the Port Enable bit. Note that when software writes a zero to this bit there may be a delay before the bit status changes to a zero. The bit status will not read as a zero until after the reset has completed. If the port is in high-speed mode after reset is complete, the host controller will automatically enable this port (e.g. set the Port Enable bit to a one). A host controller must terminate the reset and stabilize the state of the port within 2 milliseconds of software transitioning this bit from a one to a zero. For example: if the port detects that the attached device is high-speed during reset, then the host controller must have the port in the enabled state within 2ms of software writing this bit to a zero. The HCHalted bit in the USBSTS register should be a zero before software attempts to use this bit. The host controller may hold Port Reset asserted to a one when the HCHalted bit is a one.</p> <p>This field is zero if Port Power is zero.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	<p>Suspend 1'b1: Port in suspend state 1'b0: Port not in suspend state Default 0. Port Enabled Bit and Suspend bit of this register define the port states as follows. Bits [Port Enabled, Suspend] Port State 2'b0X: Disable 2'b10: Enable 2'b11: Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction, if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. A write of zero to this bit is ignored by the host controller. The host controller will unconditionally set this bit to a zero when: (1)Software sets the Force Port Resume bit to a zero (from a one). (2)Software sets the Port Reset bit to a one (from a zero). If host software sets this bit to a one when the port is not enabled (i.e. Port enabled bit is a zero) the results are undefined. This field is zero if Port Power is zero.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RW	0x0	<p>Force_Port_Resume</p> <p>1'b1: Resume detected/driven on port</p> <p>1'b0: No resume (Kstate)</p> <p>Detected / driven on port. Default 0. This functionality defined for manipulating this bit depends on the value of the Suspend bit. For example, if the port is not suspended (Suspend and Enabled bits are a one) and software transitions this bit to a one, then the effects on the bus are undefined. Software sets this bit to a 1 to drive resume signaling. The Host Controller sets this bit to a 1 if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to a one. If software sets this bit to a one, the host controller must not set the Port Change Detect bit. Note that when the EHCI controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. Software must appropriately time the Resume and set this bit to a zero when the appropriate amount of time has elapsed. Writing a zero (from one) causes the port to return to high-speed mode (forcing the bus below the port into a high-speed idle). This bit will remain a one until the port has switched to the high-speed idle. The host controller must complete this transition within 2 milliseconds of software setting this bit to a zero. This field is zero if Port Power is zero.</p>
5	W1C	0x0	<p>Over_current_Change</p> <p>Default 0.</p> <p>1'b1: This bit gets set to a one when there is a change to Over-current Active. Software clears this bit by writing a one to this bit position.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RO	0x0	<p>Over_Current_Active Default 0.</p> <p>1'b1: This port currently has an over-current condition. 1'b0: This port does not have an over-current condition. This bit will automatically transition from a one to a zero when the over current condition is removed.</p>
3	W1C	0x0	<p>Port_Ena_Chng Port Enable/Disable Change 1'b1: Port enabled/disabled status has changed. 1'b0: No change.</p> <p>Default 0. For the root hub, this bit gets set to a one only when a port is disabled due to the appropriate conditions existing at the EOF2 point. Software clears this bit by writing a 1 to it. This field is zero if Port Power is zero.</p>
2	RW	0x0	<p>Port_Ena Port Enabled/Disabled 1'b1: Enable 1'b0: Disable.</p> <p>Default 0. Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. The host controller will only set this bit to a one when the reset sequence determines that the attached device is a high-speed device. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by host software.</p> <p>Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled (0b) downstream propagation of data is blocked on this port, except for reset.</p> <p>This field is zero if Port Power is zero.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	W1C	0x0	Connect_Sts_Chng Connect Status Change 1'b1: Change in Current Connect Status 1'b0: No change Default 0. Indicates a change has occurred in the port's Current Connect Status. The host controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be "setting" an already-set bit (i.e., the bit will remain set). Software sets this bit to 0 by writing a 1 to it. This field is zero if Port Power is zero
0	RO	0x0	Cur_Connect_Sts Current Connect Status 1'b1: Device is present on port 1'b0: No device is present 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 zero if Port Power is zero.

**HSIC\_INSNREG00**

Address: Operational Base + offset (0x0090)

Programmable Microframe Base Value

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13:1	RW	0x0000	microframe_counter [13:1]: This value is used as the 1-microframe counter with byte interface (8-bits). [12:1]: This value is used as the 1-microframe counter with word interface (16-bits).
0	RW	0x0	REG00_Ena REG00 Enable Writing 1'b1 enables this register. Note: Do not enable this register for the gate-level netlist.

**HSIC\_INSNREG01**

Address: Operational Base + offset (0x0094)  
Programmable Packet Buffer OUT/IN Thresholds

Bit	Attr	Reset Value	Description
31:16	RW	0x0000	OUT_Threshold The value specified here is the number of DWORDs 32-bit entries. The OUT threshold is used to start the USB transfer as soon as the OUT threshold amount of data is fetched from system memory. It is also used to disconnect the data fetch, if the threshold amount of space is not available in the Packet Buffer.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	<p>IN_Threshold The value specified here is the number of DWORDs 32-bit entries. The IN threshold is used to start the memory transfer as soon as the IN threshold amount of data is available in the Packet Buffer. It is also used to disconnect the data write, if the threshold amount of data is not available in the Packet Buffer. The minimum OUT and IN threshold amount that can be programmed through INSN registers is 16 bytes. For INCRX configurations, the minimum threshold amount that can be programmed is the highest possible INCRX burst value. For example, if the value of the strap signals ss_ena_incr16_i, ss_ena_incr8_i, ss_ena_incr4_i is 3'b011 (for example, INCR16 burst is disabled, INCR8/INCR4 bursts are enabled), then the minimum OUT and IN threshold value should be 32 bytes (8 DWords).</p> <p>OUT and IN threshold values can be equal to the packet buffer depth only when one of the following conditions is met:</p> <ul style="list-style-type: none"> <li>(1)Packet buffer depth is equal to 512 bytes and isochronous/interrupt transactions are not initiated by the host controller.</li> <li>(2)Packet buffer depth is equal to 1024 bytes. The default value of thresholds depend on one of the following packet buffer configurations</li> </ul> <ul style="list-style-type: none"> <li>(1)1024 bytes depth, 256 bytes IN and OUT thresholds</li> <li>(2)512 bytes depth, 128 bytes IN and OUT thresholds</li> <li>(3)256 bytes depth, 64 bytes IN and OUT thresholds</li> <li>(4)128 bytes depth, 64 bytes IN and OUT thresholds</li> <li>(5)64 bytes depth, 60 bytes IN and OUT thresholds</li> </ul>

**HSIC\_INSNREG02**

Address: Operational Base + offset (0x0098)

Programmable Packet Buffer Depth

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:0	RW	0x000	Pack_Buf_Depth Programmable Packet Buffer Depth The value specified here is the number of DWORDs (32-bit entries). For a maximum 256 entries for 1 Kb packet buffer, bits [8:0] are sufficient.

**HSIC\_INSNREG03**

Address: Operational Base + offset (0x009c)

Register0000 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13	RW	0x0	Ignore_Linestate Ignore Linestate during TestSE0 Nak When set to 1 (default), the core ignores the linestate checking when transmitting SOF during the SE0_NAK test mode. When Set to 0, the port state machine disables the port if it does not find the linestate to be in SE0 when transmitting SOF during the SE0_NAK testing. While doing impedance measurement during the SE0_NAK testing, the linestate could go to non SE0 forcing the core to disable the port. This bit is used to control the port behavior during this.
12:10	RW	0x0	Delay_Add Tx-Tx turnaround Delay Add on This field specifies the extra delays in phy_clks to be added to the "Transmit to Transmit turnaround delay" value maintained in the core. The default value of this register field is 0. This default value of 0 is sufficient for most PHYs. But for some PHYs which puts wait states during the token packet, it may be required to program a value greater than 0 to meet the transmit to transmit minimum turnaround time. The recommendation to use the default value of 0 and change it only if there is an issue with minimum transmit-to-transmit turnaround time. This value should be programmed during core initialization and should not be changed afterwards.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	<p>PF_List_Fetch Periodic Frame List Fetch In CONFIG1 mode only ("EHCI Descriptor/Data Prefetching" is disabled in core configuration), setting this bit will force the host controller to fetch the periodic frame list in every micro-frame of a frame. If not set, then the periodic frame list will be fetched only in micro-frame 0 of every frame. The default is 0 (not set). This bit can be changed only during core initialization and should not be changed afterwards.</p>
8:1	RW	0x00	<p>TAO Time Available Offset This value indicates the additional number of bytes to be accommodated for the time-available calculation. The USB traffic on the bus can be started only when sufficient time is available to complete the packet within the EOF1 point. Refer to the USB 2.0 specification for details of the EOF1 point. This time-available calculation is done in the hardware, and can be further offset by programming a value in this location. Note: Time-available calculation is added for future flexibility. The application is not required to program this field by default.</p>
0	RW	0x0	<p>Break_Mem_Trans_Ena Break Memory Transfer (in CONFIG1 mode only, not applicable in CONFIG2 mode) 1'b1: Enables this function 1'b0: Disables this function Used in conjunction with INSNREG01 to enable breaking memory transactions into chunks once the OUT/IN threshold value is reached. Note: The default value for INSNREG03[0] is actually dependent on host core configuration. So if INCRx support is enabled, then this bit will be 1 after reset. Otherwise, it should stay at 0.</p>

**HSIC\_INSNREG04**

Address: Operational Base + offset (0x00a0)

Register0000 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x0	Auto_Mask automatic feature disable 1'b0: 0 by default, the automatic feature is enabled. The Suspend signal is de-asserted (logic level 1'b1) when run/stop is reset by software, but the hchalted bit is not yet set. 1'b1: Disables the automatic feature, which takes all ports out of suspend when software clears the run/stop bit. This is for backward compatibility. Bit [5] has an added functionality. For systems where the host is halted without waking up all ports out of suspend, the port can become stuck because the PHYCLK is not running when the halt is programmed. To avoid this, the DWC H20AHB host core automatically pulls ports out of suspend when the host is halted by software. This bit is used to disable this automatic function. Reset value is 1'b0.
4	RW	0x0	NakReloadFixDis NAK reload fix disable 1'b0: NAK reload fix enabled. 1'b1: NAK reload fix disabled. (Incorrect NAK reload transition at the end of a micro-frame for backward compatibility with Release 2.40c. For more information see the USB 2.0 Host-AHB Release Notes. Reset value is 1'b0.)
3	RO	0x0	reserved
2	RW	0x0	scale_down 1'b1: Scales down port enumeration time.
1	RW	0x0	HCCPARAM_WR The HCCPARAMS register's bits 17, 15:4, and 2:0 become writable. Upon system reset, these bits are 0.
0	RW	0x0	HCSPARAM_WR 1'b1: The HCSPARAMS register becomes writable. Upon system reset, this bit is 0.

**HSIC\_INSNREG05**

Address: Operational Base + offset (0x00a4)

Control and Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:18	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17	RW	0x0	VBusy Hardware indicator that a write to this register has occurred and the hardware is currently processing the operation defined by the data written. When processing is finished, this bit is cleared.
16:13	RW	0x0	VPort Field0000 Abstract Valid values range from 1 to 15. For example, if the number of ports is 3, then software should only write values 1, 2, and 3 to this field and not any other values in the range, that is, 0 or 4 -15. Suppose the software writes value 4 to VPort, from that write onwards, any writes to this register are ignored and the read value will always be 4.
12	RW	0x0	VControlLoadM 1'b0: Load 1'b1: NOP
11:8	RW	0x0	VControl
7:0	RO	0x00	VStatus

**HSIC\_INSNREG06**

Address: Operational Base + offset (0x00a8)

AHB Error Status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	Err_Capture AHB Error Captured Indicator that an AHB error was encountered and values were captured. To clear this field the application must write a 0 to it.
30:11	RO	0x0	reserved
10:8	RO	0x0	Hburst HBURST Value of the control phase at which the AHB error occurred.
7:4	RO	0x0	Expect_Beat_Num Field0000 Abstract Number of beats expected in the burst at which the AHB error occurred. Valid values are 0 to 16. 5'b10001-5'b11111: Reserved 5'b00000-5'b10000: Valid

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RO	0x0	Success_Beats_Num Number of successfully-completed beats in the current burst before the AHB error occurred. These fields apply only to AHB INCRX enabled configurations.

**HSIC\_INSNREG07**

Address: Operational Base + offset (0x00ac)

AHB Master Error Address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	Err_Address AHB Master Error Address AHB address of the control phase at which the AHB error occurred

**HSIC\_INSNREG08**

Address: Operational Base + offset (0x00b0)

HSIC Enable/Disable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x0	PORT5_HSIC_ENA PORT5 in HSIC Enabled state enable 1'b0: When HSIC configuration is selected, then a value of 0 on this control register bit will put PORT5 in the HSIC Disabled state 1'b1: When HSIC configuration is selected, then a value of 1 on this control register bit will put PORT5 in the HSIC enabled state
4	RW	0x0	PORT4_HSIC_ENA PORT4 in HSIC Enabled state enable 1'b0: When HSIC configuration is selected, then a value of 0 on this control register bit will put PORT4 in the HSIC Disabled state 1'b1: When HSIC configuration is selected, then a value of 1 on this control register bit will put PORT4 in the HSIC enabled state
3	RW	0x0	PORT3_HSIC_ENA PORT3 in HSIC Enabled state enable 1'b0: When HSIC configuration is selected, then a value of 0 on this control register bit will put PORT3 in the HSIC Disabled state 1'b1: When HSIC configuration is selected, then a value of 1 on this control register bit will put PORT3 in the HSIC enabled state

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x0	PORT2_HSIC_ENA PORT2 in HSIC Enabled state enable 1'b0: When HSIC configuration is selected, then a value of 0 on this control register bit will put PORT2 in the HSIC Disabled state 1'b1: When HSIC configuration is selected, then a value of 1 on this control register bit will put PORT2 in the HSIC enabled state
1	RW	0x0	PORT1_HSIC_ENA PORT1 in HSIC Enabled state enable 1'b0: When HSIC configuration is selected, then a value of 0 on this control register bit will put PORT1 in the HSIC Disabled state 1'b1: When HSIC configuration is selected, then a value of 1 on this control register bit will put PORT1 in the HSIC enabled state
0	RW	0x0	PORT0_HSIC_ENA PORT0 in HSIC Enabled state enable 1'b0: When HSIC configuration is selected, then a value of 0 on this control register bit will put PORT0 in the HSIC Disabled state 1'b1: When HSIC configuration is selected, then a value of 1 on this control register bit will put PORT0 in the HSIC enabled state

## 21.6 Application Note

1. Set "incr\_x\_en" to 1, and set these signals "incr16\_en, incr8\_en, incr4" to 1 or 0 as you need, to increase the performance of transfer. Otherwise, the default burst type would be "SINGLE".

## **Chapter 22 HOST Interface**

Will update soon!

Rockchip Confidential

## Chapter 23 Crypto

### 23.1 Overview

Crypto is a hardware accelerator of encrypting or decrypting. It supports the most commonly used algorithm: DES/3DES, AES, SHA1, SHA256, MD5 and RSA.

The Crypto supports following features:

- Support AES 128/192/256 bits key mode, ECB/CBC/CTR chain mode, Slave/FIFO mode
- Support DES/3DES (ECB and CBC chain mode), 3DES (EDE/ EEE key mode), Slave/FIFO mode
- Support SHA1/SHA256/MD5 (with hardware padding) HASH function, FIFO mode only
- Support 160 bit Pseudo Random Number Generator (PRNG)
- Support PKA 512/1024/2048 bit Exp Modulator
- Support up to 150M clock frequency

### 23.2 Block Diagram

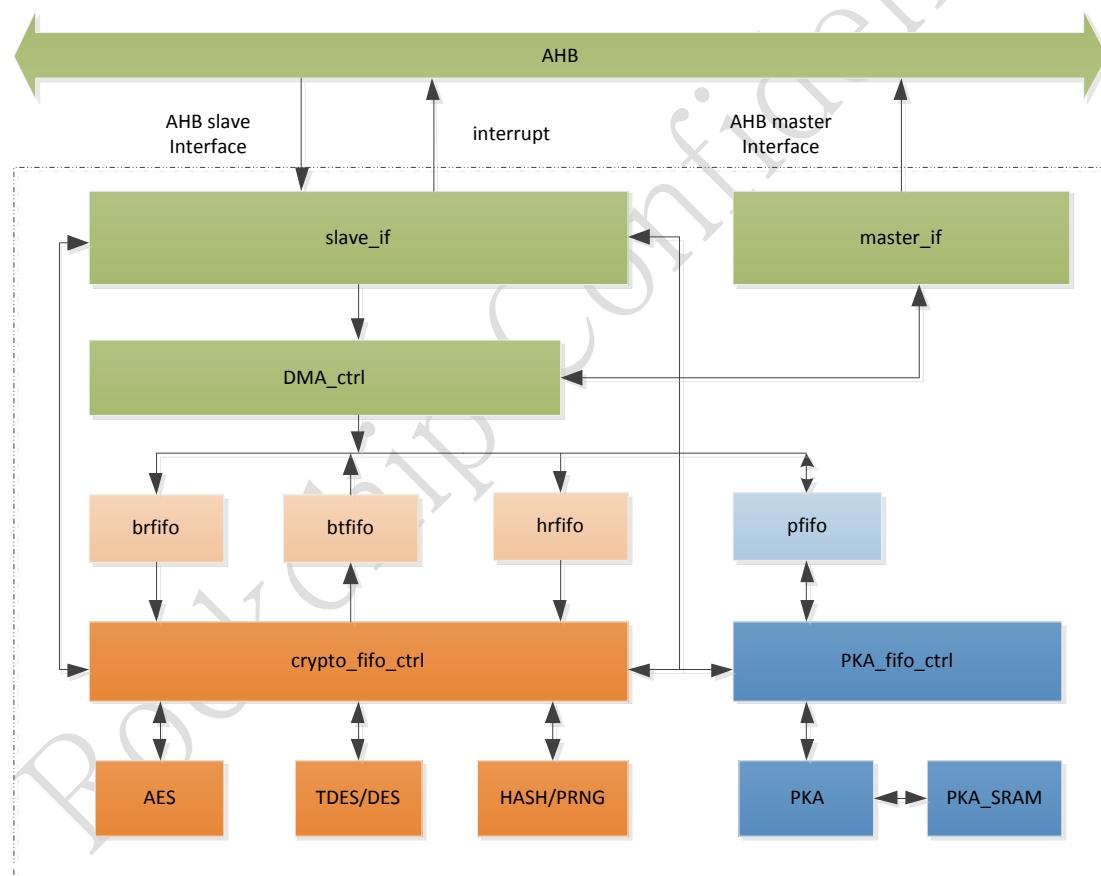


Fig. 23-1 Crypto Architecture

Figure above shows the architecture of Crypto.

## 23.3 Register description

### 23.3.1 Register Summary

Name	Offset	Size	Reset Value	Description
CRYPTO_INTSTS	0x0000	W	0x00000000	Interrupt Status Register
CRYPTO_INTENA	0x0004	W	0x00000000	Interrupt Set Register
CRYPTO_CTRL	0x0008	W	0x00000000	Control Register
CRYPTO_CONF	0x000c	W	0x00000000	
CRYPTO_BRDMAS	0x0010	W	0x00000000	Block Receiving DMA Start Address Register
CRYPTO_BTDMAS	0x0014	W	0x00000000	Block Transmiting DMA Start Address Register
CRYPTO_BRDMAL	0x0018	W	0x00000000	Block Receiving DMA Length Register
CRYPTO_HRDMAS	0x001c	W	0x00000000	Hash Receiving DMA Start Address Register
CRYPTO_HRDMAL	0x0020	W	0x00000000	Hash Receiving DMA Length Register
CRYPTO_AES_CTRL	0x0080	W	0x00000000	AES Control Register
CRYPTO_AES_STS	0x0084	W	0x00000000	Status Register
CRYPTO_AES_DIN_0	0x0088	W	0x00000000	AES Input Data 0 Register
CRYPTO_AES_DIN_1	0x008c	W	0x00000000	AES Input Data 1 Register
CRYPTO_AES_DIN_2	0x0090	W	0x00000000	AES Input Data 2 Register
CRYPTO_AES_DIN_3	0x0094	W	0x00000000	AES Input Data 3 Register
CRYPTO_AES_DOUT_0	0x0098	W	0x00000000	AES Output Data 0 Register
CRYPTO_AES_DOUT_1	0x009c	W	0x00000000	AES Output Data 1 Register
CRYPTO_AES_DOUT_2	0x00a0	W	0x00000000	AES Output Data 2 Register
CRYPTO_AES_DOUT_3	0x00a4	W	0x00000000	AES Output Data 3 Register
CRYPTO_AES_IV_0	0x00a8	W	0x00000000	AES IV data 0 Register
CRYPTO_AES_IV_1	0x00ac	W	0x00000000	AES IV data 1 Register
CRYPTO_AES_IV_2	0x00b0	W	0x00000000	AES IV data 2 Register
CRYPTO_AES_IV_3	0x00b4	W	0x00000000	AES IV data 3 Register
CRYPTO_AES_KEY_0	0x00b8	W	0x00000000	AES Key data 0 Register
CRYPTO_AES_KEY_1	0x00bc	W	0x00000000	AES Key data 1 Register
CRYPTO_AES_KEY_2	0x00c0	W	0x00000000	AES Key data 2 Register
CRYPTO_AES_KEY_3	0x00c4	W	0x00000000	AES Key data 3 Register
CRYPTO_AES_KEY_4	0x00c8	W	0x00000000	AES Key data 4 Register
CRYPTO_AES_KEY_5	0x00cc	W	0x00000000	AES Key data 5 Register
CRYPTO_AES_KEY_6	0x00d0	W	0x00000000	AES Key data 6 Register

Name	Offset	Size	Reset Value	Description
CRYPTO_AES_KEY_7	0x00d4	W	0x00000000	AES Key data 7 Register
CRYPTO_AES_CNT_0	0x00d8	W	0x00000000	AES Input Counter 0 Register
CRYPTO_AES_CNT_1	0x00dc	W	0x00000000	AES Input Counter 1 Register
CRYPTO_AES_CNT_2	0x00e0	W	0x00000000	AES Input Counter 2 Register
CRYPTO_AES_CNT_3	0x00e4	W	0x00000000	AES Input Counter 3 Register
CRYPTO_TDES_CTRL	0x0100	W	0x00000000	TDES Control Register
CRYPTO_TDES_STS	0x0104	W	0x00000000	Status Register
CRYPTO_TDES_DIN_0	0x0108	W	0x00000000	TDES Input Data 0 Register
CRYPTO_TDES_DIN_1	0x010c	W	0x00000000	TDES Input Data 1 Register
CRYPTO_TDES_DOU_T_0	0x0110	W	0x00000000	TDES Output Data 0 Register
CRYPTO_TDES_DOU_T_1	0x0114	W	0x00000000	TDES Output Data 1 Register
CRYPTO_TDES_IV_0	0x0118	W	0x00000000	TDES IV data 0 Register
CRYPTO_TDES_IV_1	0x011c	W	0x00000000	TDES IV data 1 Register
CRYPTO_TDES_KEY1_0	0x0120	W	0x00000000	TDES Key1 data 1 Register
CRYPTO_TDES_KEY1_1	0x0124	W	0x00000000	TDES Key1 data 1 Register
CRYPTO_TDES_KEY2_0	0x0128	W	0x00000000	TDES Key2 data 0 Register
CRYPTO_TDES_KEY2_1	0x012c	W	0x00000000	TDES Key2 data 1 Register
CRYPTO_TDES_KEY3_0	0x0130	W	0x00000000	TDES Key3 data 0 Register
CRYPTO_TDES_KEY3_1	0x0134	W	0x00000000	TDES Key3 data 1 Register
CRYPTO_HASH_CTRL	0x0180	W	0x00000000	Hash Control Register
CRYPTO_HASH_STS	0x0184	W	0x00000000	Hash Status Register
CRYPTO_HASH_MSG_LEN	0x0188	W	0x00000000	Hash Message Len
CRYPTO_HASH_DOU_T_0	0x018c	W	0x00000000	Hash Result Register 0
CRYPTO_HASH_DOU_T_1	0x0190	W	0x00000000	Hash Result Register 1
CRYPTO_HASH_DOU_T_2	0x0194	W	0x00000000	Hash Result Register 2
CRYPTO_HASH_DOU_T_3	0x0198	W	0x00000000	Hash Result Register 3
CRYPTO_HASH_DOU_T_4	0x019c	W	0x00000000	Hash Result Register 4

Name	Offset	Size	Reset Value	Description
CRYPTO_HASH_DOUT_5	0x01a0	W	0x00000000	Hash Result Register 4
CRYPTO_HASH_DOUT_6	0x01a4	W	0x00000000	Hash Result Register 6
CRYPTO_HASH_DOUT_7	0x01a8	W	0x00000000	Hash Result Register 7
CRYPTO_HASH_SEED_0	0x01ac	W	0x00000000	PRNG Seed/HMAC Key Register 0
CRYPTO_HASH_SEED_1	0x01b0	W	0x00000000	PRNG Seed/HMAC Key Register 1
CRYPTO_HASH_SEED_2	0x01b4	W	0x00000000	PRNG Seed/HMAC Key Register 2
CRYPTO_HASH_SEED_3	0x01b8	W	0x00000000	PRNG Seed/HMAC Key Register 3
CRYPTO_HASH_SEED_4	0x01bc	W	0x00000000	PRNG Seed/HMAC Key Register 4
CRYPTO_TRNG_CTRL	0x0200	W	0x00000000	TRNG Control
CRYPTO_TRNG_DOUT_0	0x0204	W	0x00000000	TRNG Output Data 0
CRYPTO_TRNG_DOUT_1	0x0208	W	0x00000000	TRNG Output Data 1
CRYPTO_TRNG_DOUT_2	0x020c	W	0x00000000	TRNG Output Data 2
CRYPTO_TRNG_DOUT_3	0x0210	W	0x00000000	TRNG Output Data 3
CRYPTO_TRNG_DOUT_4	0x0214	W	0x00000000	TRNG Output Data 4
CRYPTO_TRNG_DOUT_5	0x0218	W	0x00000000	TRNG Output Data 5
CRYPTO_TRNG_DOUT_6	0x021c	W	0x00000000	TRNG Output Data 6
CRYPTO_TRNG_DOUT_7	0x0220	W	0x00000000	TRNG Output Data 7
CRYPTO_PKA_CTRL	0x0280	W	0x00000000	PKA Control Register
CRYPTO_PKA_M	0x0400	W	0x00000000	
CRYPTO_PKA_C	0x0500	W	0x00000000	
CRYPTO_PKA_N	0x0600	W	0x00000000	
CRYPTO_PKA_E	0x0700	W	0x00000000	

Notes: **Size** : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

### 23.3.2 Detail Register Description

#### CRYPTO\_INTSTS

Address: Operational Base + offset (0x0000)

Interrupt Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x0	PKA_DONE_INT PKA Done Interrupt
4	W1C	0x0	HASH_DONE_INT Hash Done Interrupt
3	W1C	0x0	HRDMA_ERR_INT Specifies the interrupt of hash receiving DMA Error
2	W1C	0x0	HRDMA_DONE_INT Specifies the interrupt of hash receiving DMA DONE
1	W1C	0x0	BCDMA_ERR_INT Specifies the interrupt of block cipher Error
0	W1C	0x0	BCDMA_DONE_INT Specifies the interrupt of block cipher DONE

**CRYPTO\_INTENA**

Address: Operational Base + offset (0x0004)

Interrupt Set Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x0	PKA_DONE_ENA Set the interrupt Enable of PKA done 1'b1: enable 1'b0: disable
4	RW	0x0	HASH_DONE_ENA Set the interrupt Enable of hash done 1'b1: enable 1'b0: disable
3	RW	0x0	HRDMA_ERR_ENA Set the interrupt Enable of hash receiving DMA Error 1'b1: enable 1'b0: disable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x0	HRDMA_DONE_ENA Set the interrupt Enable of hash receiving DMA DONE 1'b1: enable 1'b0: disable
1	RW	0x0	BCDMA_ERR_ENA Set the interrupt Enable of block cipher DMA Error 1'b1: enable 1'b0: disable
0	RW	0x0	BCDMA_DONE_ENA Set the interrupt Enable of block cipher DMA DONE 1'b1: enable 1'b0: disable

**CRYPTO\_CTRL**

Address: Operational Base + offset (0x0008)

Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	Write_Mask
15:10	RO	0x0	reserved
9	RW	0x0	TRNG_FLUSH FLUSH TRNG Software write 1 to start. When finishes, the core will clear it.
8	RWSC	0x0	TRNG_START Start TRNG Software write 1 to start. When finishes, the core will clear it.
7	RWSC	0x0	PKA_FLUSH Software write 1 to start Flush Process. The process will clear BRFIFO, BTFIFO, and state machine. Then Software should write 0 to end FLUSH Process
6	RW	0x0	HASH_FLUSH Software write 1 to start Flush Process. The process will clear BRFIFO, BTFIFO, and state machine. Then Software should write 0 to end FLUSH Process

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	BLOCK_FLUSH Software write 1 to start Flush Process. The process will clear BRFIFO, BTFIFO, and state machine. Then Software should write 0 to end FLUSH Process. It must last for at least 20 cycles to clean registers and FSM
4	RWSC	0x0	PKA_START Starts/initializes PKA Software write 1 to start. When finishes, the core will clear it.
3	RWSC	0x0	HASH_START Starts/initializes HASH/PRNG/HMAC Software write 1 to start. When finishes, the core will clear it.
2	RWSC	0x0	BLOCK_START Starts/initializes Block Cipher Software write 1 to start. When finishes, the core will clear it.
1	RWSC	0x0	TDES_START Starts/initializes TDES Software write 1 to start. When finishes, the core will clear it.
0	RWSC	0x0	AES_START Starts/initializes AES Software write 1 to start. When finishes, the core will clear it. Software can also write 0 to clear it.

**CRYPTO\_CONF**

Address: Operational Base + offset (0x000c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8	RW	0x0	HR_ADDR_MODE Hash Receive DMA Address Mode 1'b1: fix 1'b0: increment
7	RW	0x0	BT_ADDR_MODE Block Transmit DMA Address Mode 1'b1: fix 1'b0: increment
6	RW	0x0	BR_ADDR_MODE Block Receive DMA Address Mode 1'b1: fix 1'b0: increment

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	Byteswap_HRFIFO If this bit is high, then the data read from the bus is byte-swapped in a word boundary. If this bit is low (default), then the data is handed over to the FIFO without byte-swap. For little endian bus, this bit should be 1'b1.
4	RW	0x0	Byteswap_BTFIFO If this bit is high, then the data read from the bus is byte-swapped in a word boundary. If this bit is low (default), then the data is handed over to the FIFO without byte-swap. For little endian bus, this bit should be 1'b1.
3	RW	0x0	Byteswap_BRFIFO If this bit is high, then the data read from the bus is byte-swapped in a word boundary. If this bit is low (default), then the data is handed over to the FIFO without byte-swap. For little endian bus, this bit should be 1'b1.
2	RW	0x0	DESSEL Specifies the Destination block cipher of FIFO. AES(=0)/DES(=1)
1:0	RW	0x0	HASHINSEL Specifies the following Data from independent source (0) Data from block cipher input (1) Data from block cipher output (2) Reserved (3)

**CRYPTO\_BRDMAS**

Address: Operational Base + offset (0x0010)

Block Receiving DMA Start Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	STARTADDR Specifies the Start Address of DMA The address should be aligned by 32-bit.

**CRYPTO\_BTDMAS**

Address: Operational Base + offset (0x0014)

Block Transmitting DMA Start Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	STARTADDR Specifies the Start Address of DMA The address needs to be aligned by 32-bit.

**CRYPTO\_BRDMAL**

Address: Operational Base + offset (0x0018)

Block Receiving DMA Length Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	LENGTH Specifies the Block length of DMA. The length unit is WORD.

**CRYPTO\_HRDMAS**

Address: Operational Base + offset (0x001c)

Hash Receiving DMA Start Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	STARTADDR Specifies the Start Address of DMA The address needs to be aligned by 32-bit.

**CRYPTO\_HRDMAL**

Address: Operational Base + offset (0x0020)

Hash Receiving DMA Length Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	LENGTH Specifies the Block length of DMA. The length unit is BYTE.

**CRYPTO\_AES\_CTRL**

Address: Operational Base + offset (0x0080)

AES Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11	RW	0x0	AES_BitSwap_CNT Change the Big-endian and Little-endian by swapping the byte order. 0 = Disables Counter data byte swap 1 = Enables Counter data byte swap
10	RW	0x0	AES_BitSwap_Key Change the Big-endian and Little-endian by swapping the byte order. 0 = Disables Key byte swap 1 = Enables Key byte swap
9	RW	0x0	AES_BitSwap_IV Change the Big-endian and Little-endian by swapping the byte order. 0 = Disables Initial value byte swap 1 = Enables Initial value byte swap

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	RW	0x0	AES_ByteSwap_DO Change the Big-endian and Little-endian by swapping the byte order. 0 = Disables Output data byte swap 1 = Enables Output data byte swap
7	RW	0x0	AES_ByteSwap_DI Change the Big-endian and Little-endian by swapping the byte order. 0 = Disables Input data byte swap 1 = Enables Input data byte swap
6	RW	0x0	AES_KeyChange Specifies the AES key change mode selection signal. When the bit is asserted, it will not do key-expansion function to calculate new sub-key. So it is a faster way, when several times of calculation use the same key. But if the keys are different, asserting this bit will have the wrong result. 0 = Key is not changed 1 = Key is changed
5:4	RW	0x0	AES_ChainMode Specifies AES chain mode selection 00 = ECB mode 01 = CBC mode 10 = CTR mode
3:2	RW	0x0	AES_KeySize Specifies the AES key size selection signal 00 : 128-bit key 01 : 192-bit key 10 : 256-bit key
1	RW	0x0	AES_FifoMode Specify AES Fifo Mode 1'b0: Slave mode 1'b1: fifo mode
0	RW	0x0	AES_Enc Specifies the Encryption/ Decryption mode selection signal 0 : Encryption 1 : Decryption

**CRYPTO\_AES\_STS**

Address: Operational Base + offset (0x0084)

Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	AES_DONE When AES finish, it will be HIGH, And it will not be LOW until it restart . 1: done 0: not done

**CRYPTO\_AES\_DIN\_0**

Address: Operational Base + offset (0x0088)

AES Input Data 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_DIN_0 Specifies AES Input data [127:96].

**CRYPTO\_AES\_DIN\_1**

Address: Operational Base + offset (0x008c)

AES Input Data 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_DIN_1 Specifies AES Input data [95:64].

**CRYPTO\_AES\_DIN\_2**

Address: Operational Base + offset (0x0090)

AES Input Data 2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_DIN_2 Specifies AES Input data [63:32]

**CRYPTO\_AES\_DIN\_3**

Address: Operational Base + offset (0x0094)

AES Input Data 3 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_DIN_3 Specifies AES Input data [31:0]

**CRYPTO\_AES\_DOUT\_0**

Address: Operational Base + offset (0x0098)

AES Output Data 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	AES_DOUT_0 Specifies AES Output data [127:96].

**CRYPTO\_AES\_DOUT\_1**

Address: Operational Base + offset (0x009c)

AES Output Data 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	AES_DOUT_1 Specifies the Output data [95:64].

**CRYPTO\_AES\_DOUT\_2**

Address: Operational Base + offset (0x00a0)

AES Output Data 2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	AES_DOUT_2 Specifies AES Output data [63:32].

**CRYPTO\_AES\_DOUT\_3**

Address: Operational Base + offset (0x00a4)

AES Output Data 3 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	AES_DOUT_3 Specifies AES Output data [31:0].

**CRYPTO\_AES\_IV\_0**

Address: Operational Base + offset (0x00a8)

AES IV data 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_IV_0 Specifies AES Initialization vector [127:96]

**CRYPTO\_AES\_IV\_1**

Address: Operational Base + offset (0x00ac)

AES IV data 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_IV_1 Specifies AES Initialization vector [95:64]

**CRYPTO\_AES\_IV\_2**

Address: Operational Base + offset (0x00b0)

AES IV data 2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_IV_2 Specifies AES Initialization vector [63:32]

**CRYPTO\_AES\_IV\_3**

Address: Operational Base + offset (0x00b4)

AES IV data 3 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_IV_3 Specifies AES Initialization vector [31:0]

**CRYPTO\_AES\_KEY\_0**

Address: Operational Base + offset (0x00b8)

AES Key data 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_KEY_0 Specifies AES key data [255:224]

**CRYPTO\_AES\_KEY\_1**

Address: Operational Base + offset (0x00bc)

AES Key data 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_KEY_1 Specifies AES key data [223:192]

**CRYPTO\_AES\_KEY\_2**

Address: Operational Base + offset (0x00c0)

AES Key data 2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_KEY_2 Specifies AES key data [191:160]

**CRYPTO\_AES\_KEY\_3**

Address: Operational Base + offset (0x00c4)

AES Key data 3 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_KEY_3 Specifies AES key data [159:128]

**CRYPTO\_AES\_KEY\_4**

Address: Operational Base + offset (0x00c8)

AES Key data 4 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_KEY_4 Specifies AES key data [127:96]

**CRYPTO\_AES\_KEY\_5**

Address: Operational Base + offset (0x00cc)

AES Key data 5 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_KEY_5 Specifies the key data [95:64]

**CRYPTO\_AES\_KEY\_6**

Address: Operational Base + offset (0x00d0)

AES Key data 6 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_KEY_6 Specifies AES key data [63:32]

**CRYPTO\_AES\_KEY\_7**

Address: Operational Base + offset (0x00d4)

AES Key data 7 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_KEY_7 Specifies the key data [31:0]

**CRYPTO\_AES\_CNT\_0**

Address: Operational Base + offset (0x00d8)

AES Input Counter 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_CNT_0 Specifies AES Input Counter [127:96].

**CRYPTO\_AES\_CNT\_1**

Address: Operational Base + offset (0x00dc)

AES Input Counter 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_CNT_1 Specifies AES Input Counter [95:64].

**CRYPTO\_AES\_CNT\_2**

Address: Operational Base + offset (0x00e0)

AES Input Counter 2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_CNT_2 Specifies AES Input Counter[63:32]

**CRYPTO\_AES\_CNT\_3**

Address: Operational Base + offset (0x00e4)

AES Input Counter 3 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_CNT_3 Specifies AES Input Counter [31:0]

**CRYPTO\_TDES\_CTRL**

Address: Operational Base + offset (0x0100)

TDES Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8	RW	0x0	TDES_BitSwap_Key 0 = Disables Key byte swap 1 = Enables Key byte swap
7	RW	0x0	TDES_BitSwap_IV 0 = Disables Initial value byte swap 1 = Enables Initial value byte swap
6	RW	0x0	TDES_BitSwap_DO 0 = Disables Output data byte swap 1 = Enables Output data byte swap
5	RW	0x0	TDES_BitSwap_DI 0 = Disables Input data byte swap 1 = Enables Input data byte swap
4	RW	0x0	TDES_ChainMode Specifies TDES chain mode selection 0 : ECB mode 1 : CBC mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	TDES_EEE Specifies the TDES key mode selection 1'b0 : EDE 1'b1 : EEE
2	RW	0x0	TDES_Select Specify DES or TDES cipher 1'b0 : DES 1'b1 : TDES
1	RW	0x0	TDES_FifoMode Specify TDES Fifo Mode 1'b0: Slave mode 1'b1: Fifo mode
0	RW	0x0	TDES_Enc Specifies the Encryption/ Decryption mode selection signal 0 : Encryption 1 : Decryption

**CRYPTO\_TDES\_STS**

Address: Operational Base + offset (0x0104)

Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	TDES_DONE When DES/TDES finishes, it will be HIGH, And it will not be LOW until it restart . 1: done 0: not done

**CRYPTO\_TDES\_DIN\_0**

Address: Operational Base + offset (0x0108)

TDES Input Data 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	TDES_DIN_0 Specifies TDES Input data [63:32].

**CRYPTO\_TDES\_DIN\_1**

Address: Operational Base + offset (0x010c)

TDES Input Data 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	TDES_DIN_1 Specifies TDES Input data [31:0].

**CRYPTO\_TDES\_DOUT\_0**

Address: Operational Base + offset (0x0110)

TDES Output Data 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	TDES_DOUT_0 Specifies TDES Output data [63:32].

**CRYPTO\_TDES\_DOUT\_1**

Address: Operational Base + offset (0x0114)

TDES Output Data 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	TDES_DOUT_1 Specifies TDES Output data [31:0].

**CRYPTO\_TDES\_IV\_0**

Address: Operational Base + offset (0x0118)

TDES IV data 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	TDES_IV_0 Specifies TDES Initialization vector [63:32]

**CRYPTO\_TDES\_IV\_1**

Address: Operational Base + offset (0x011c)

TDES IV data 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	TDES_IV_1 Specifies TDES Initialization vector [31:0]

**CRYPTO\_TDES\_KEY1\_0**

Address: Operational Base + offset (0x0120)

TDES Key1 data 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	TDES_KEY1_0 Specifies TDES key1 data [63:32]

**CRYPTO\_TDES\_KEY1\_1**

Address: Operational Base + offset (0x0124)

TDES Key1 data 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	TDES_KEY1_1 Specifies TDES key1 data [31:0]

**CRYPTO\_TDES\_KEY2\_0**

Address: Operational Base + offset (0x0128)

TDES Key2 data 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	TDES_KEY2_0 Specifies TDES key2 data [63:32]

**CRYPTO\_TDES\_KEY2\_1**

Address: Operational Base + offset (0x012c)

TDES Key2 data 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	TDES_KEY_1 Specifies TDES key data [31:0]

**CRYPTO\_TDES\_KEY3\_0**

Address: Operational Base + offset (0x0130)

TDES Key3 data 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	TDES_KEY3_0 Specifies TDES key3 data [63:32]

**CRYPTO\_TDES\_KEY3\_1**

Address: Operational Base + offset (0x0134)

TDES Key3 data 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	AES_KEY3_1 Specifies TDES key3 data [31:0]

**CRYPTO\_HASH\_CTRL**

Address: Operational Base + offset (0x0180)

Hash Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	HASH_SWAP_DO Specifies the Byte swap of data output (hash result) 0 = Does not swap (default) 1 = Swap
2	RW	0x0	HASH_SWAP_DI Specifies the Byte swap of data input. 0 = Does not swap (default) 1 = Swap
1:0	RW	0x0	Engine_Selection 2'b00: SHA1_HASH 2'b01: MD5_HASH 2'b10: SHA256_HASH 2'b11: PRNG

**CRYPTO\_HASH\_STS**

Address: Operational Base + offset (0x0184)

Hash Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	HASH_DONE Hash Done Signal When HASH finishes, it will be HIGH, And it will not be LOW until it restart 1'b1 : done 1'b0 : not done

**CRYPTO\_HASH\_MSG\_LEN**

Address: Operational Base + offset (0x0188)

Hash Message Len

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	Msg_size Hash total byte.

**CRYPTO\_HASH\_DOUT\_0**

Address: Operational Base + offset (0x018c)

Hash Result Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HASH_RESULT_0 Specifies the HASH Result [159:128]

**CRYPTO\_HASH\_DOUT\_1**

Address: Operational Base + offset (0x0190)

Hash Result Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HASH_RESULT_1 Specifies the HASH Result [127:96]

**CRYPTO\_HASH\_DOUT\_2**

Address: Operational Base + offset (0x0194)

Hash Result Register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HASH_RESULT_2 Specifies the HASH Result [95:64]

**CRYPTO\_HASH\_DOUT\_3**

Address: Operational Base + offset (0x0198)

Hash Result Register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HASH_RESULT_3 Specifies the HASH Result [63:32]

**CRYPTO\_HASH\_DOUT\_4**

Address: Operational Base + offset (0x019c)

Hash Result Register 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HASH_RESULT_4 Specifies the HASH Result [31:0]

**CRYPTO\_HASH\_DOUT\_5**

Address: Operational Base + offset (0x01a0)

Hash Result Register 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HASH_RESULT_5 Specifies the HASH Result [31:0]

**CRYPTO\_HASH\_DOUT\_6**

Address: Operational Base + offset (0x01a4)

Hash Result Register 6

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HASH_RESULT_6 Specifies the HASH Result [31:0]

**CRYPTO\_HASH\_DOUT\_7**

Address: Operational Base + offset (0x01a8)

Hash Result Register 7

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HASH_RESULT_7 Specifies the HASH Result [31:0]

**CRYPTO\_HASH\_SEED\_0**

Address: Operational Base + offset (0x01ac)

PRNG Seed/HMAC Key Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	HASH_SEED_0 Specifies PRNG Seed/HMAC Key buffer [159:128]

**CRYPTO\_HASH\_SEED\_1**

Address: Operational Base + offset (0x01b0)

PRNG Seed/HMAC Key Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	HASH_SEED_1 Specifies PRNG Seed/HMAC Key buffer [127:96]

**CRYPTO\_HASH\_SEED\_2**

Address: Operational Base + offset (0x01b4)

PRNG Seed/HMAC Key Register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	HASH_SEED_2 Specifies PRNG Seed/HMAC Key buffer [95:64]

**CRYPTO\_HASH\_SEED\_3**

Address: Operational Base + offset (0x01b8)

PRNG Seed/HMAC Key Register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	HASH_SEED_3 Specifies PRNG Seed/HMAC Key buffer [63:32]

**CRYPTO\_HASH\_SEED\_4**

Address: Operational Base + offset (0x01bc)

## PRNG Seed/HMAC Key Register 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	HASH_SEED_4 Specifies PRNG Seed/HMAC Key buffer [31:0]

**CRYPTO\_TRNG\_CTRL**

Address: Operational Base + offset (0x0200)

TRNG Control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved
16	RW	0x0	osc_enable osc_ring enable It control the running of osc_ring. And it is independent of clock and flush signal. This means that it can run even when clock is gating or flush is asserted as long as osc_enable is asserted. Before it is used to get TRNG result, please run osc_ring first to get enough entropy. 1'b1: Enable ; 1'b0: Disable ;
15:0	RW	0x0000	period sample period TRNG use clock_crypto to sample ring osc output, this parameter is specify how many cycles to generate 1 bit random data.

**CRYPTO\_TRNG\_DOUT\_0**

Address: Operational Base + offset (0x0204)

TRNG Output Data 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	TRNG_DOUT_0

**CRYPTO\_TRNG\_DOUT\_1**

Address: Operational Base + offset (0x0208)

TRNG Output Data 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	TRNG_DOUT_1

**CRYPTO\_TRNG\_DOUT\_2**

Address: Operational Base + offset (0x020c)

TRNG Output Data 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	TRNG_DOUT_2

**CRYPTO\_TRNG\_DOUT\_3**

Address: Operational Base + offset (0x0210)

TRNG Output Data 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	TRNG_DOUT_3

**CRYPTO\_TRNG\_DOUT\_4**

Address: Operational Base + offset (0x0214)

TRNG Output Data 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	TRNG_DOUT_4

**CRYPTO\_TRNG\_DOUT\_5**

Address: Operational Base + offset (0x0218)

TRNG Output Data 5

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	TRNG_DOUT_5

**CRYPTO\_TRNG\_DOUT\_6**

Address: Operational Base + offset (0x021c)

TRNG Output Data 6

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	TRNG_DOUT_6

**CRYPTO\_TRNG\_DOUT\_7**

Address: Operational Base + offset (0x0220)

TRNG Output Data 7

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	TRNG_DOUT_7

**CRYPTO\_PKA\_CTRL**

Address: Operational Base + offset (0x0280)

PKA Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RW	0x0	block_size PKA Size It specifies the bits of N in PKA calculation. 2'b00: 512 bit 2'b01: 1024 bit 2'b10: 2048 bit

**CRYPTO\_PKA\_M**

Address: Operational Base + offset (0x0400)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	m PKA input or output data. PKA result = (M ^ E) mod N. When it finishes, the result data is in M position. Start from PKA_M base address, and may contain 512/1024/2048 bits data.

**CRYPTO\_PKA\_C**

Address: Operational Base + offset (0x0500)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	c PKA pre-calculate data, C = 2 ^ (2n+2) mod N

**CRYPTO\_PKA\_N**

Address: Operational Base + offset (0x0600)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	n PKA modular

**CRYPTO\_PKA\_E**

Address: Operational Base + offset (0x0700)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	e PKA exponent.

## 23.4 Application Note

### 23.4.1 Reset a port

CRU\_SOFTRST3\_CON.crypto\_srstn\_req is used to do a soft reset to crypto . Please refer to "Chapter CRU" for more details.

### 23.4.2 Overall Performance

Use CLKSEL24\_CON.crypto\_div\_con to select crypto frequency:  $F_{\text{crypto}} = F_{\text{aclk}} / (\text{div} + 1)$ .  
Make sure  $F_{\text{crypto}}$  do not exceed 150M.

The performance of crypto FIFO mode is list below.

<b>algorithm</b>	<b>cycle</b>	<b>block size</b>	<b>frequency</b>	<b>throughput rate</b>
DES	17	64 bit	100M	376 M bps
TDES	51	64 bit	100M	125 M bps
AES	11/13/15	128 bit	100M	1160/984/853Mbps
SHA-1	81	512 bit	100M	632 Mbps
MD5	65	512 bit	100M	787 Mbps

### 23.4.3 Usage

#### 1. Symmetric algorithm

DES/3DES, AES are symmetric algorithms. There are two ways of using these algorithms:  
Slave mode and FIFO mode.

In Slave mode, you can calculate 1 block size of data by starting the engine. Take AES-128 for example, you should

- Program Input 128 bit Data to AES\_DIN\_0~AES\_DIN\_3
- Program Input 128 bit Key to AES\_KEY\_0~AES\_KEY\_3
- Program control mode to AES\_CTRL to run in different mode
- Program CTRL.AES\_START to run
- wait AES\_STS.DONE High
- Read AES\_DOUT\_0 ~ AES\_DOUT\_3 to get result.

In FIFO mode,

- Program the source address to BRDMAS, the destination address to BTDMAS, program the length in word unit to BRDMAL;
- Program Input 128 bit Key to AES\_KEY\_0~AES\_KEY\_3;
- Program control mode to AES\_CTRL to run in different mode;
- Program INTENA to enable interrupt;
- Program CTRL.BLOCK\_START to start;
- wait interrupt asserted;
- Program INTSTS to clear interrupt status;
- Read the destination address which BTDMA points to.

FIFO mode get much higher throughput rate.

#### 2. HASH

HASH is used to get digest of data. Only support FIFO mode.

There are three source: (1) hr\_fifo; (2) br\_fifo; (3) bt\_fifo.

Take hr\_fifo for example

- (1) Program CTRL.HASH\_FLUSH 1'b1 to clear, wait several cycle ( $\geq 10$  cycles), and Program CTRL.HASH\_FLUSH 1'b0
- (2) Program data source address to HRDMAS, program 1 time data length in word unit to HRDMAL, program total length in byte unit to HASH\_MSG\_LEN
- (3) Program HASH\_CTRL to choose algorithm, for example SHA-256
- (4) Program INTENA to enable interrupt;
- (5) Program CTRL.HASH\_START 1'b1 to start;
- (6) Wait interrupt asserted; Only if HRDMAL length meets can this interrupt be asserted
- (7) If you have another section of data to hash, then go to (2), HASH\_MSG\_LEN need not to be programmed;  
else go to (8)
- (8) wait HASH\_STS.done asserted. Only if Hash\_MSG\_LEN meet can this bit status register asserted.
- (9) Read HASH\_DOUT\_0 – HASH\_DOUT\_7 to get result.

### 3. Asymmetric Algorithm

Support 512/1024/2048 bit RSA calculation. It provide the big number calculation. Result =  $M^E \bmod N$

Program CTRL.PKA\_FLUSH 1'b1 to flush RSA module;

Wait CTRL.PKA\_FLUSH to be LOW. It is self-cleared;

Program input\_data(M) to PKA\_M; Program pre\_caculated C to PKA\_C; Program Key(N) to PKA\_N; Program Key(E) to PKA\_E.  $C = 2^{(2n+2)} \bmod N$ . n is the required bit of N. For example 2048 bit N, n = 2048;

Program PKA\_CTRL to select RSA size: 512/1024/2048

Program INTENA to enable interrupt;

Program CTRL.PKA\_START to start;

Wait interrupt asserted.

Read PKA\_M to get results.

## Chapter 24 Graphics Processing Unit (GPU)

### 24.1 Overview

With support for 2D graphics, 3D graphics, and GPGPU computing, the GPU Series of GPUs provides a complete graphics acceleration platform based on open standards.

The GPU brings graphics power to a wide range of different products, from mobile user interfaces to tablets, HDTV, and mobile gaming. One single driver stack for all graphics configurations simplifies application porting, system integration, and maintenance. Scheduling and performance scaling are fully handled within the graphics system with no special considerations required from the application developer.

The GPU takes graphics instructions from software applications running on the application processor. It processes the graphics instructions and puts the results back into system memory. These results are sent to the framebuffer and on to the display device.

GPU supports the following features:

- Fully programmable architecture
- A rich API feature set with high-performance support for both shader-based and fixed-function graphics APIs.
- Anti-aliasing capabilities
- An effective core for General Purpose computing on GPU(GPGPU) applications
- High memory bandwidth and low power consumption for 3D graphics content
- Scalability for products from smart phones to high-end mobile computing
- Performance leading 3D graphics
- Image quality using double-precision FP64, and anti-aliasing
- Standard bus interfaces
- An ACE-Lite interface to access external memory
- Easy integration
- Latency tolerance Configurable per-core power management for enabling the optimal power and performance combination for each application
- Coherency aware interconnects for system memory and resource sharing
- Frame buffer compression
- Supported API graphics industry standards
  - OpenGL ES 1.1 Specification at Khronos
  - OpenGL ES 2.0 Specification at Khronos
  - OpenGL ES 3.0 Specification at Khronos
  - OpenCL 1.0 Specification at Khronos, Full Profile specification
  - OpenCL 1.1 Specification at Khronos, Full Profile specification
  - DirectX 9 Specification
  - DirectX 11.1 Specification

GPU performance showed as below:

- Triangle rate: 325Mtri/s
- Fill rate: 2600Mpix/s
- FP32: 88GFLOPS

## 24.2 Block Diagram

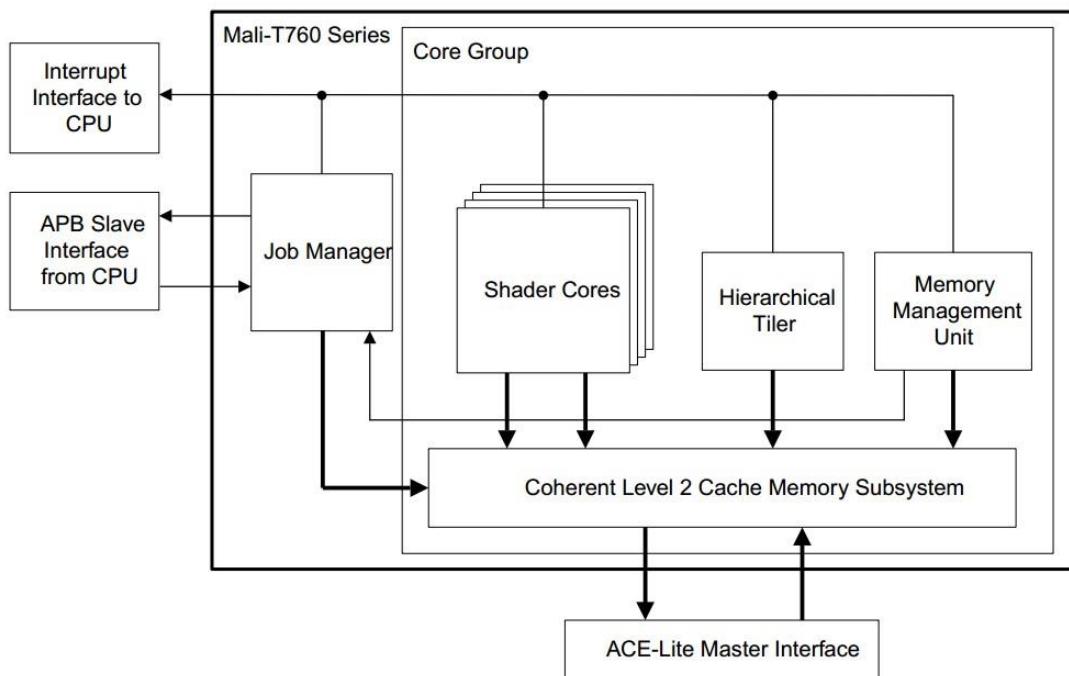


Fig. 24-1 GPU Block Diagram

## 24.3 Function Description

Please refer to the document  
[mali\\_t760\\_r0p0\\_technical\\_reference\\_manual\\_100006\\_0000\\_00\\_en.pdf](mali_t760_r0p0_technical_reference_manual_100006_0000_00_en.pdf).

## 24.4 Register Description

The GPU address range is 0xFFA300-0xFFA400.

Please refer to the document  
[mali\\_t760\\_r0p0\\_technical\\_reference\\_manual\\_100006\\_0000\\_00\\_en.pdf](mali_t760_r0p0_technical_reference_manual_100006_0000_00_en.pdf).

## 24.5 Interface Description

Please refer to the document  
[mali\\_t760\\_r0p0\\_technical\\_reference\\_manual\\_100006\\_0000\\_00\\_en.pdf](mali_t760_r0p0_technical_reference_manual_100006_0000_00_en.pdf).

## 24.6 Application Notes

### 24.6.1 Clock and Reset control

We can config CRU\_CLKSEL34\_CON[7:6] to select GPU AXI clock source among CPLL/GPLL/usbphy(480MHz)/NPLL, and then set CRU\_CLKSEL34\_CON[4:0] to divide clock source from 1:1 to 1:32.

We can config CRU\_CLKGATE5\_CON[7] to gate the gpu clock and config CRU\_CLKGATE11\_CON[13] to gate the gpu AXI clock.

We can reset gpu core by setting CRU\_SOFTRST7\_CON[8].

## **24.6.2 GPU interrupt**

The GPU generates interrupts for job handling, memory management, and events not tied to a specific shader core. For these, the number of logical interrupts is individually controlled, but there is only a single physical interrupt line for each group.

The following top level interrupts are raised by the GPU:

- The GPU\_IRQGPU (GIC interrupt 40)

Exceptions that are not associated with specific jobs. These cannot be recovered.

- GPU\_IRQJOB (GIC interrupt 38)

Signals the completion or failure of a job running on the GPU.

- GPU\_IRQMMU (GIC interrupt 39)

Exceptions caused by memory management. These are potentially recoverable.

The application processor interprets the interrupts generated by the GPU . The software queries the states of the registers in the job manager to determine what must be done to handle the interrupt. The job manager is not required to wait for the result of the interrupt, it can be starting on the next set of jobs to be executed.

## **24.6.3 Others**

Please refer to the document

[mali\\_t760\\_r0p0\\_technical\\_reference\\_manual\\_100006\\_0000\\_00\\_en.pdf](#).

# Chapter 25 Video encoder & decoder Unit (VCODEC)

## 25.1 Overview

VCODEC is composed of video decoder and video encoder. VCODEC is connected to VCODEC\_AHB bus through an AHB slave and VCODEC\_AXI through an AXI master. The register setting is configured through the AHB slave interface and the stream data is read and written through the AXI master interface.

Video decoder and video encoder share many internal memories and they also share the bus master and slave interfaces. So it prevents video decoder and video encoder from working simultaneously. Encoding and decoding now have to time-share the memory resource on a frame by frame basis.

Video decoder and encoder share the MMU (memory management unit) and AXI bus. Video decoder can support cacheable bus operation.

## 25.2 Block Diagram

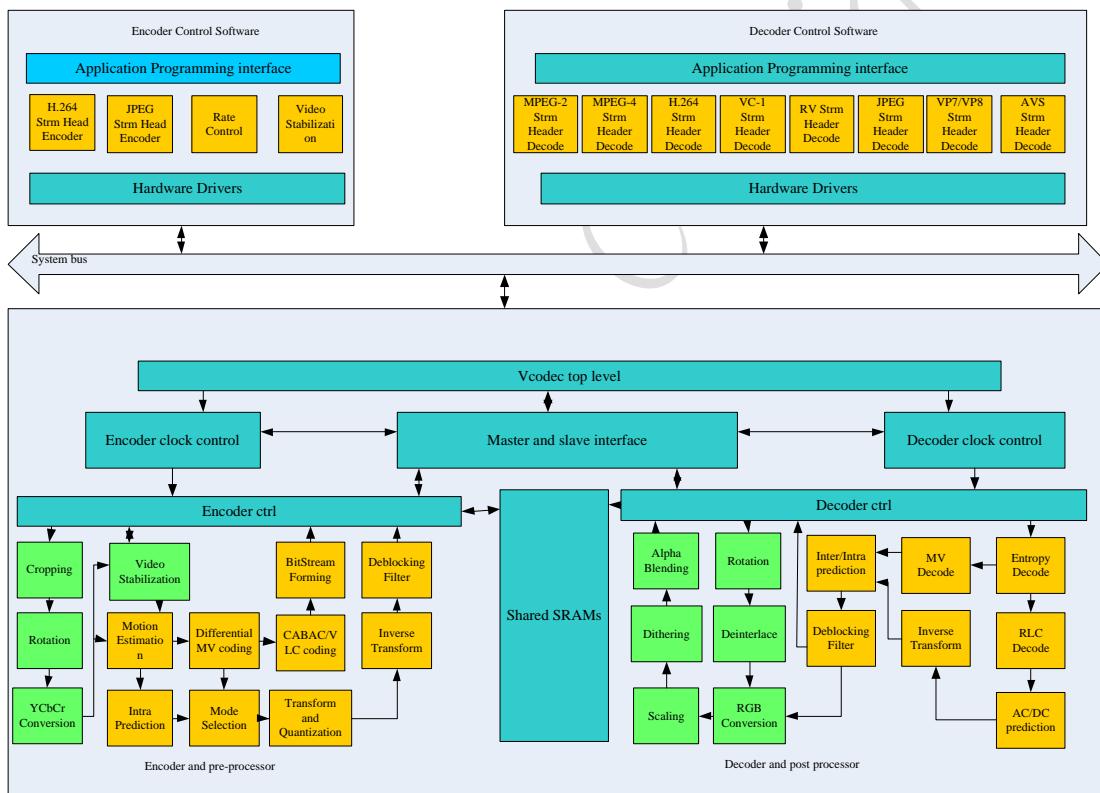


Fig. 25-1 VCODEC block diagram

The block diagram of VCODEC is shown as Fig. 25-1. The lower big box is the VCODEC hardware (HW) implementation. The left part of the lower big box is encoder and pre-processor. The green boxes in it present pre-processor part and video stabilization part while the yellow boxes in it present encoder part. Pre-processor is pipelined with the encoder and it can only be used with the encoder. Video stabilization detects and compensates undesired jitter effect on the video while the desired effects like panning are maintained. Video stabilization can be used pipelined with video encoding or in standalone mode when video encoding is disabled. Video stabilization can detect scene changes in the video sequence. Key frames are encoded when a scene change is detected. This will help improve the encoding

quality.

The right part of the lower big box is decoder and post-processor. The green boxes in it present post-processor while the yellow boxes in it present decoder. Post-processor can run in stand-alone mode or pipeline mode. In stand-alone mode, it can process image data from any external source. In pipeline mode, it can reduce bus bandwidth, because post-process can read its input data directly from the decoder output without accessing external memory.

## 25.3 Function Description

Decoder support multi-format stream decoding, as Table 25-1 shows. The decoder has a big embedded reference buffer, which can enhance the performance.

Table 25-1 Decoder supported standards, profiles and levels

Standard	Decoder support
H.264 profile and level	Baseline Profile, level 1-5.2 Main Profile, level 1-5.2 High Profile, level 1-5.2 Image size up to 2160p at level 5.2
SVC profile and level	Scalable Baseline Profile, base layer only Scalable High Profile, base layer only
MVC profile and level	Stereo High
MPEG-4 visual profile and level	Simple Profile, levels 0-6 Advanced Simple Profile, levels 0-5
MPEG-2 profile and level	Main Profile, low, medium and high levels
MPEG-1 profile and level	Main Profile, low, medium and high levels
H.263 profile and level	Profile 0, levels 10-70. Image size up to 720x576
Sorenson Spark profile and level	Bitstream version 0 and 1
VC-1 profile and level	Simple Profile, low, medium and high levels Main Profile, low, medium and high levels Advanced Profile, levels 0-3
JPEG profile and level	Baseline interleaved JPEG supports ROI (region of image) decode
RV profile and level	RV8 RV9 RV10
VP7 profile and level	VP7 version 0-3
VP8 profile and level	VP8 version2 (webM)
WebP profile and level	WebP
AVS profile and level	P2 Jizhun Profile, level 6.0 and 6.2
DivX profile and level	DivX Home Theater Profile Qualified DivX3 DivX4 DivX5 Divx6

The post-processor in stand-alone mode supports rotation, Deinterlace, RGB conversion, Scaling, Dithering and Alpha blending, it supports only RGB conversion, Scaling, Dithering and Alpha blending in pipeline mode.

Encoder supports H.264 and JPEG encoding, as Table 25-2 shows.

Table 25-2 Encoder supported standard, profile and level

Standard	Encoder support
----------	-----------------

H.264 Profile and level	Baseline Profile, levels 1-4.1 Main Profile, levels 1-4.1 High Profile, levels 1-4.1
JPEG profile and level	Baseline(DCT sequential)

The pre-processor supports cropping, rotation and YCbCr conversion.

As the decoder and encoder hardware share SRAM, they can't work simultaneously, but they can time-share the SRAM on a frame by frame basic. The decoder and encoder are both multi-instance capable. As the decoder and hardware has no information of previously encoded or decoded frames stores in internal memory or registers, the input data can be changed each time a frame is encoded or decoded. The format and resolution can be totally different from the previous one.

## 25.4 Video frame format

This chapter describes different input and output video frame formats supported by VCODEC. Each function module has its own supported video frame formats, and this chapter describes all the video frame formats.

### 25.4.1 YCbCr 4:2:0 Planar Format

In the planar format, each video sample component forms one memory plane. Within one plane, the data has to be stored linearly and contiguously in the memory as shown in Fig. 25-2. The luminance samples are stored in raster-scan order ( $Y_0 Y_1 Y_2 Y_3 Y_4 \dots$ ). The chrominance samples are stored in two planes also in raster scan order ( $Cb_0 Cb_1 Cb_2 Cb_3 Cb_4 \dots$  and  $Cr_0 Cr_1 Cr_2 Cr_3 Cr_4 \dots$ ). In this format each pixel takes 12 bits of memory.

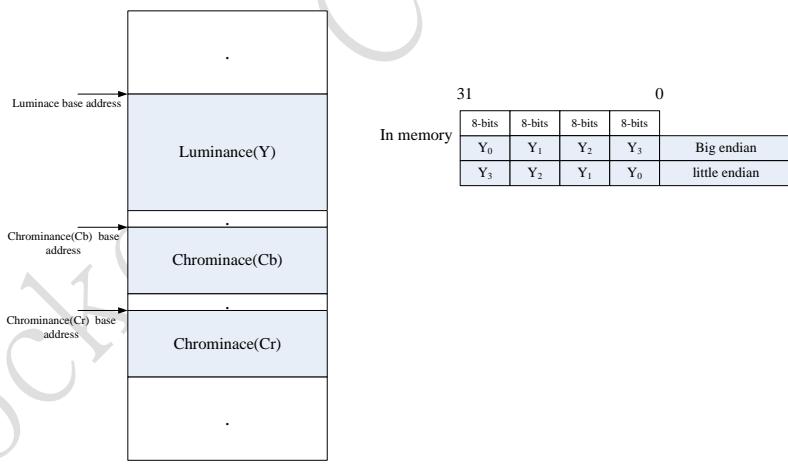


Fig. 25-2 VCODEC YCbCr 4:2:0 planar format

### 25.4.2 YCbCr 4:2:0 Semi-Planar format

In semi-planar YcbCr4:2:0 format the luminance samples from one plane in memory, and chrominance samples from another. Within one plane, the data has to be stored linearly and contiguously in the memory. The luminance pixels are stored in raster-scan order ( $Y_0 Y_1 Y_2 Y_3 Y_4 \dots$ ). The interleaved chrominance CbCr samples are stored in raster-scan order in memory as  $Cb_0 Cr_0 Cb_1 Cr_1 Cb_2 Cr_2 Cb_3 Cr_3 Cb_4 Cr_4 \dots$

Semi-Planar format supports both progressive and interlaced format as presented in Fig. 25-3. The interlaced format may be alternative line or each line.

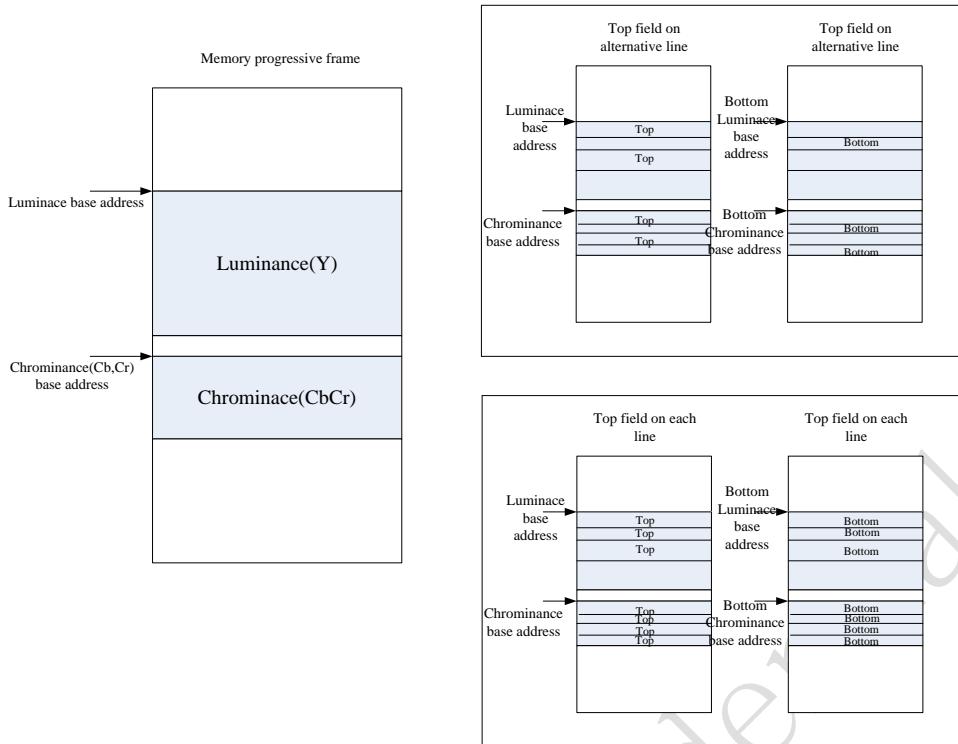


Fig. 25-3 VCODEC YCbCr 4:2:0 Semi-planar format

### 25.4.3 YCbCr 4:2:0 Tiled Semi-Planar Format

Like the YCbCr 4:2:0 semi-planar format, the tiled semi-planar format is also organized in the memory on two separate planes. The difference between these formats is that in tiled format the pixel samples are not anymore in raster-scan order but are stored macroblock(16x16 pixels) by macroblock. The samples of each macroblock are stored in consecutive addresses and the macroblocks are ordered from left to right and from top to down as Fig. 25-4. When this format used as input data format, it causes the lowest bus load to the system as there is minimal amount of non-sequential memory addressing required when reading the input data to the post-processor.

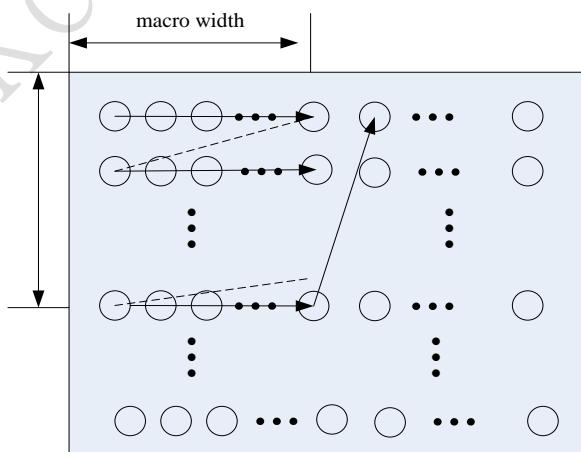


Fig. 25-4 VCODEC Tile scan mode

### 25.4.4 YCbCr 4:2:2 Interleaved Format

In the interleaved YCbCr 4:2:2 format the pixel samples from a single plane in which the data

has to be stored linearly and contiguously as shown in Fig. 25-5. The pixel data is in raster scan order and the chrominance samples are interleaved between the luminance samples as  $Y_0C_{b0} Y_1C_{r0} Y_2 C_{b1} Y_3C_{r1} Y_4 C_{b2} \dots$ . YCrCb, CbYCrY and CrYCbY component orders are supported also. In this format, each pixel takes 16 bits in the memory.

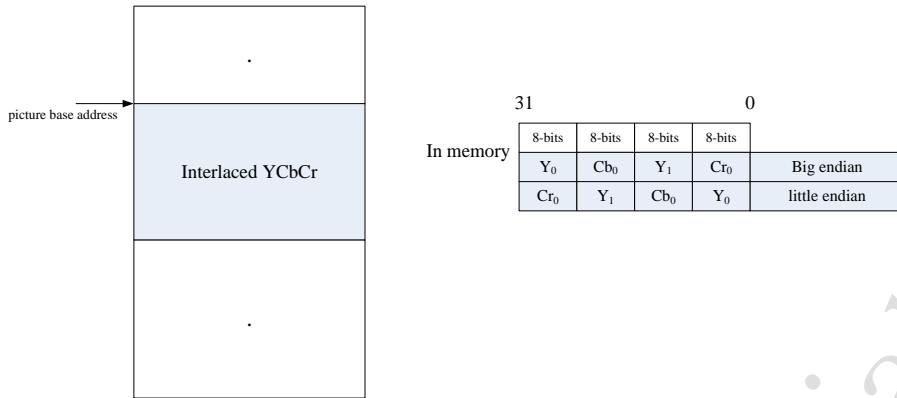


Fig. 25-5 VCODEC YCbCr4:2:2 Interleaved format

#### 25.4.5 AYCbCr 4:4:4 Interleaved Format

In the interleaved YcbCr 4:2:2 format, the pixel samples from a single plane in which the data has to be stored linearly and contiguously as show in Fig. 25-6. The pixel data is in raster scan order and the chrominance and alpha channel samples are interleaved between the luminance samples as  $A_0Y_0 C_{b0}C_{r0} A_1 Y_1 C_{b1}C_{r1} \dots$ .

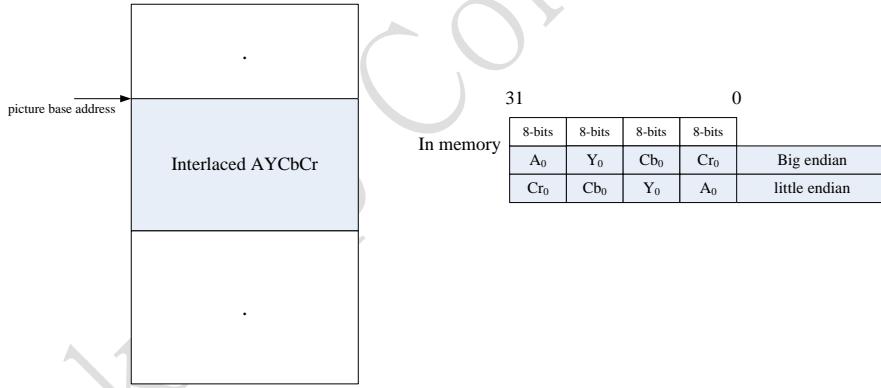


Fig. 25-6 VCODEC AYCbCr 4:4:4 Interleaved format

#### 25.4.6 RGB 16bpp Format

In this format each pixel is represented by 16 or less bits containing the red, blue and green samples. There are several 16bpp formats which use different number of bits for each sample. For example the RGB 5-5-5 format uses 5 bits for each sample and 1 bit is left unused or can represent a transparency flag, where RGB 5-6-5 uses 6 bits for the G sample and 5 bits for R and B samples. Common for all 16bpp types is that two pixels fit into one 32-bit space.

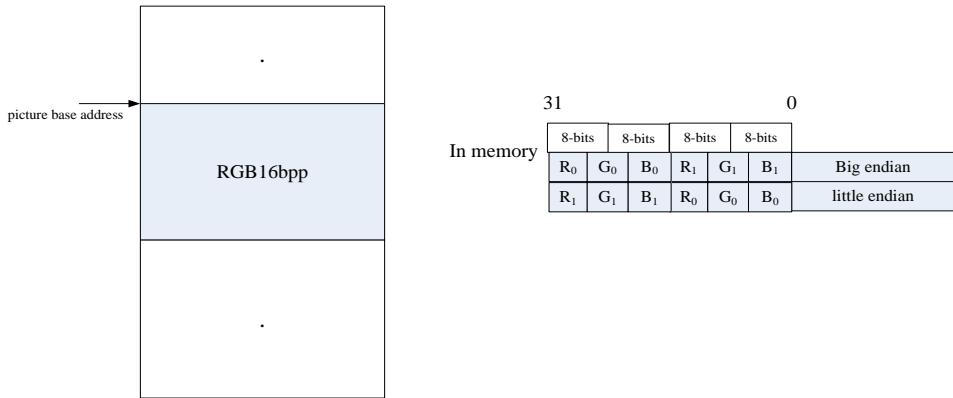


Fig. 25-7 VCODEC RGB 16bpp format

### 25.4.7 RGB 32bpp Format

Any RGB format that has its pixels represented by more than 16bits each is considered to be of 32bpp type. Typically in this format each pixel is represented by three bytes containing a red, blue and green sample and a 4<sup>th</sup> byte which can be empty or hold an alpha blending value. Common for all 32bpp types is that only one pixel fit into one 32-bit space. The data has to be stored linearly and contiguously in the memory.

## 25.5 Video Decoder

### 25.5.1 H.264 decoder

The H264 feature that Video decoder supports shows as Table 25-3.

Table 25-3 Video decoder H.264 feature

Feature	Decoder support
Input data format	H.264 byte or NAL unit stream /SVC stream /MVC stream
Decoding scheme	Frame by frame(or field by field) Slice by Slice
Output data format	YCbCr 4:2:0 semi-planar
Supported image size	48x48 to 4096x2304 Step size 16 pixels
Maximum frame rate	60fps at 1920x1088 24fps at 4096x2304
Maximum bit rate	As specified by H.264 HP level 4.2
Error detection and concealment	Supported

The input of the decoder is H.264 standard bit stream in either plain NAL unit format or byte stream format. The input format in use will be automatically detected. The H.264 video encoding allows the use of multiple reference pictures, which means that the decoding order of the pictures may be different from their display order. The decoder can perform internally the display reordering of the decoded pictures or it can skip this and output all the pictures as soon as they are decoded.

The decoder has two operating modes: in the primary mode the HW performs entropy decoding, and in the secondary mode SW performs entropy decoding. Secondary mode is used in H.264 ASO or Slice Group stream decoding.

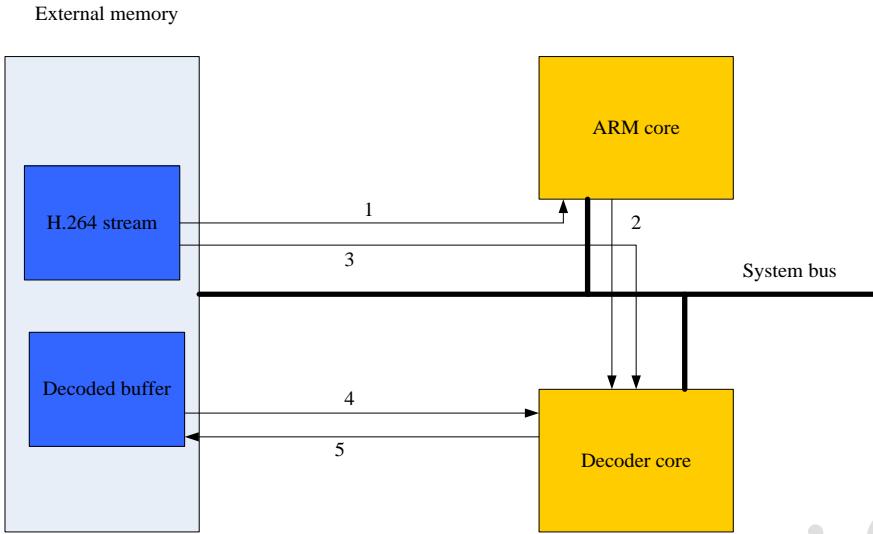


Fig. 25-8 Dataflow of HW performs entropy decoding in video decoder

The dataflow of HW performs entropy decoding is as Fig. 25-8 shown. The decoder software (SW) starts decoding the first picture by parsing the stream headers (1). Software then setups the hardware control registers (picture size, stream start address etc.) and enables the hardware (2). Hardware decodes the picture by reading stream (3) and the reference pictures (required for inter picture decoding)(4) from the external memory. Hardware writes the decoded output picture to memory one macroblock at a time (5). When the picture has been fully decoded or the hardware has run out of stream data, it gives an interrupt with a proper status flag and provides stream and address for software to continue and returns to initial state.

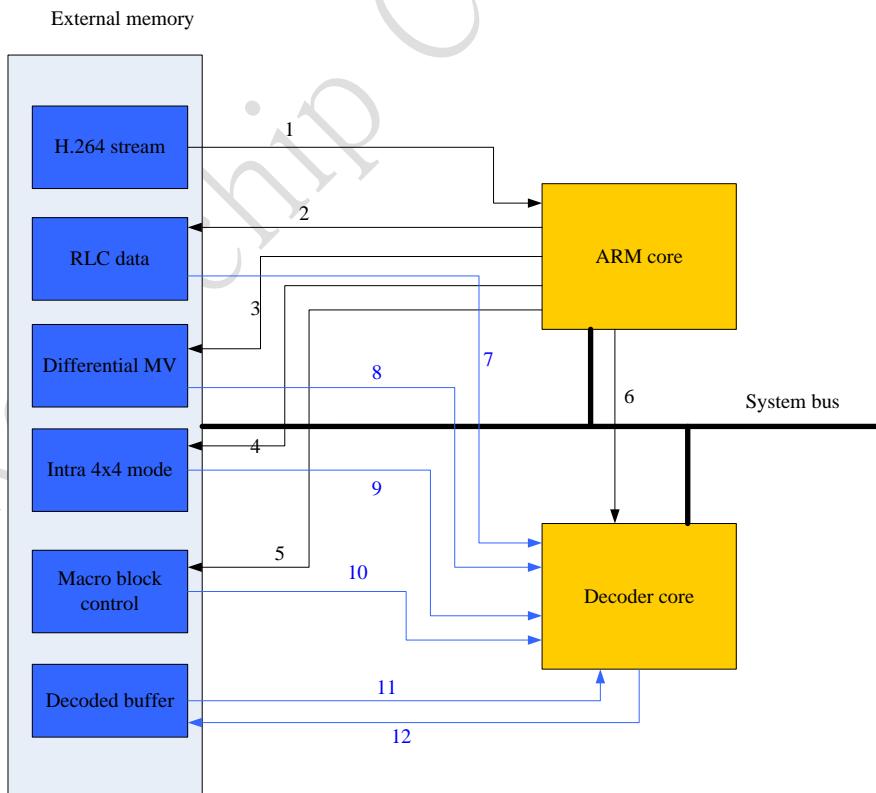


Fig. 25-9 Dataflow of SW performs entropy decoding in video decoder

SW entropy decoding mode (RLC mode) changes the input data format that is transferred

from SW to HW. The dataflow of this mode is as Fig. 25-9. In this case the decoder software starts decoding the first picture by parsing the stream headers (1), and by performing entropy decoding. Software then writes the following items to external memory:

Run-length-code (RLC) data (2)

Differential motion vectors (3)

Intra 4x4 prediction modes (4)

Macroblock control data (5)

Last step for the software is to write the hardware control registers and to enable the hardware (6).

Hardware decodes the picture by buffering control data for several macroblocks at a time, and reading then appropriate amount of RLC data, differential motion vectors and intra modes depending on each macroblock type (7)-(10). For the rest of the decoding process (11)-(12), the functionality is identical to the HW entropy decoding mode. When the picture has been fully decoded, hardware can raise an interrupt and write the status bits in the status register.

### **25.5.2 MPEG-4/H.263/SORENSEN SPARK decoder**

The features that video decoder supports about MPEG-4/H.263/SORENSEN SPAR shows as Fig. 25-4.

Table 25-4 MPEG-4/H.263/SORENSEN SPAR feature

<b>Feature</b>	<b>Decoder support</b>
Input data format	MPEG-4/H.263/Sorenson Spark elementary video stream
Decoding scheme	Frame by frame(or field by field) Video packet by video packet
Output data format	YCbCr 4:2:0 semi-planar
Supported image size	48x48 to 1920x1088(MPEG-4, Sorenson Spark) 48x48 to 720x576(H.263) Step size 16 pixels
Maximum frame rate	60fps at 1920x1088
Maximum bit rate	As specified by MPEG-4 ASP level5
Error detection and concealment	Supported

The decoder of MPEG-4/H.263/Sorenson has two operating modes: in the primary mode the HW performs entropy decoding, and in the secondary mode SW performs entropy decoding. Secondary mode is used in MPEG-4 data partitioned stream decoding.

### **25.5.3 MPEG-2/MPEG-1 decoder**

The features of MPEG-2/MPEG-1 supported by decoder are shown as Table 25-5.

Table 25-5 MPEG-2/MPEG-1 features

<b>Feature</b>	<b>Decoder support</b>
Input data format	MPEG-2/MPEG-1 elementary video stream
Decoding scheme	Frame by frame(or field by field) Video packet by video packet
Output data format	YCbCr 4:2:0 semi-planar
Supported image size	48x48 to 1920x1088 Step size 16 pixels

	MPEG-2 can support up to 3840x2160
Maximum frame rate	60fps at 1920x1088
Maximum bit rate	As specified by MPEG-2 MP high level
Error detection and concealment	Supported

The dataflow of MPEG-2/MPEG-1 is the same of H.264 HW performs entropy decoding as Fig. 25-8.

#### 25.5.4 VC-1 decoder

The features of VC-1 supported by decoder are shown as Table 25-6.

Table 25-6 VC-1 features

Feature	Decoder support
Input data format	VC-1
Decoding scheme	Frame by frame(or field by field) Slice by slice
Output data format	YCbCr 4:2:0 semi-planar
Supported image size	48x48 to 1920x1088 Step size 16 pixels
Maximum frame rate	30fps at 1920x1088
Maximum bit rate	As specified by VC-1 AP level3
Error detection and concealment	Supported

The VC-1 decoder has only one operating mode in which the HW performs entropy decoding.

#### 25.5.5 RV decoder

RV features supported by decoder are as shown in Table 25-7.

Table 25-7 RV features

Feature	Decoder support
Input data format	RV8,RV9 or RV10 stream
Decoding scheme	Frame by frame Slice by slice
Output data format	YCbCr 4:2:0 semi-planar
Supported image size	48x48 to 1920x1088 Step size 16 pixels
Maximum frame rate	60fps at 1920x1088
Maximum bit rate	As specified by RV specification
Error detection and concealment	Supported

The RV decoder has only one operating mode in which the HW performs entropy decoding.

#### 25.5.6 VP6/VP8 decoder

VP6/VP8 features supported by decoder are as shown in Table 25-8.

Table 25-8 VP6/VP8 features

Feature	Decoder support
Input data format	VP6.0/VP6.1/VP6.2/VP8 stream

Decoding scheme	Frame by frame
Output data format	YCbCr 4:2:0 semi-planar
Supported image size	48x48 to 1920x1088 Step size 16 pixels VP8 can support up to 3840x2160
Maximum frame rate	60fps at 1920x1088
Maximum bit rate	As specified by VP6/VP8 specification
Error detection and concealment	Supported

### 25.5.7 AVS decoder

AVS features supported by decoder are as shown in Table 25-9.

Table 25-9 AVS features

Feature	Decoder support
Input data format	AVS stream
Decoding scheme	Frame by frame, field by field Slice by slice
Output data format	YCbCr 4:2:0 semi-planar
Supported image size	48x48 to 1920x1088 Step size 16 pixels
Maximum frame rate	60fps at 1920x1088
Maximum bit rate	As specified by AVS standard
Error detection and concealment	Supported

### 25.5.8 DIVX decoder

DIVX features supported by decoder are as shown in Table 25-10.

Table 25-10 Divx features

Feature	Decoder support
Input data format	Divx 3,4,5 or 6 stream
Decoding scheme	Frame by frame Video packet by video packet
Output data format	YCbCr 4:2:0 semi-planar
Supported image size	48x48 to 1920x1088 Step size 16 pixels
Maximum frame rate	60fps at 1920x1088
Maximum bit rate	As specified by the Divx specification
Error detection and concealment	Supported

## 25.6 JPEG Decoder

JPEG features supported by decoder are as shown in Table 25-11.

Table 25-11 JPEG features

Feature	Decoder support
Input data format	JFIF file format 1.02 YCbCr 4:0:0, 4:2:0, 4:2:2, 4:4:0, 4:1:1 and 4:4:4

	semi-planar
Decoding scheme	Input: buffer by buffer, from 5Kb to 8MB at a time <sup>①</sup> Output: from 1 MB row to 16 Mpixels at a time <sup>②</sup>
Output data format	YCbCr 4:0:0, 4:2:0, 4:2:2, 4:4:0, 4:1:1 and 4:4:4 semi-planar
Supported image size	48x48 to 8176x8176(66.8 Mpixels) Step size 8 pixels <sup>③</sup>
Maximum frame rate	Up to 76 million pixels pre second
Maximum bit rate	As specified by the Divx specification
Thumbnail decoding	JPEG compressed thumbnails supported
Error detection	Supported

①Programmable buffer size for optimizing performance and memory consumption. Interrupt will be issued when buffer runs empty, and the control software will load more streams to external memory.

②Programmable output slice for optimizing performance and memory consumption. Interrupt will be issued when the requested area decoded. The control software can be used to switch the decoder output address each time.

③Non-16x16 dividable resolutions will be filled to 16 pixel boundary.

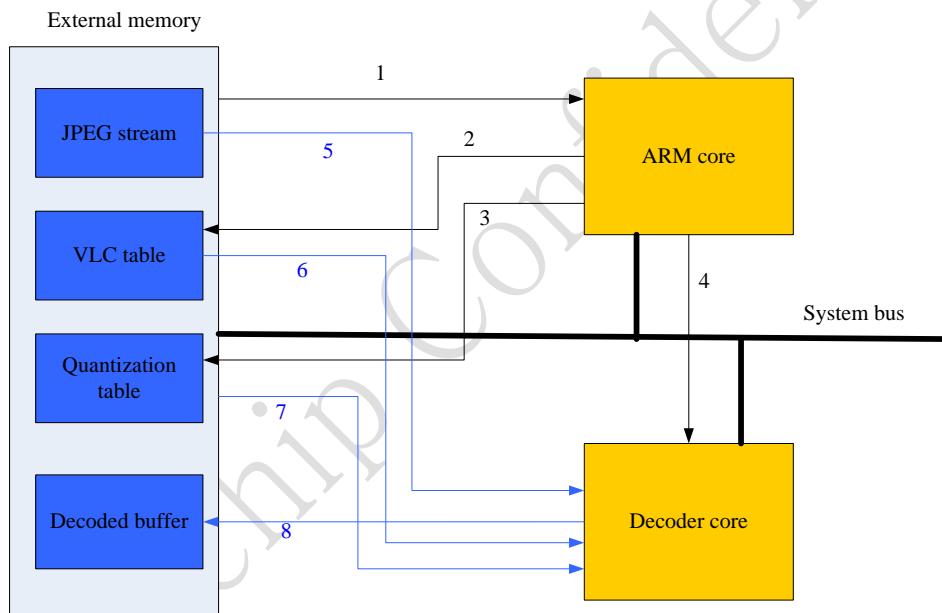


Fig. 25-10 The dataflow of JPEG decoder

The data flow of jpeg decoder is as Fig. 25-10 shown. The decoder software starts decoding the picture by parsing the stream headers (1) and then writes the following items to external memory:

VLC tables (2)

Quantization tables (3)

Last step for the software is to write the hardware control registers and to enable the hardware (4). After starting hardware, SW waits interrupt from HW.

Hardware decodes the picture by reading stream (5), VLC (6) and QP (7) tables. Hardware writes the decoded output picture memory one macroblock at a time (8). When the picture has been fully decoded, or the hardware has run out of stream data, it gives an interrupt with a proper status flag and provides stream end address for software to continue and returns to initial state.

## 25.7 Image Post-processor

The features supported by Post-processor are as show in Table 25-12.

Table 25-12 Post-processor features

Feature	Post-processor support
Input data format	Any format generated by the decoder in combined mode YCbCr 4:2:0 semi-planar YCbCr 4:2:0 planar YCbYCr 4:2:2 YCrYCb 4:2:2 CbYCrY 4:2:2 CrYCbY 4:2:2
Post-processor scheme	Frame by frame. Post-processor handles the image macroblock by macroblock, also in standalone mode.
Input image source	Internal source(combined mode) External source(standalone mode): e.g. a software decoder or camera interface
Output data format	YCbCr 4:2:0 semi-planar YCbCr 4:2:2 YCrYCb 4:2:2 CbYCrY 4:2:2 CrYCbY 4:2:2 Fully configurable ARGB channel lengths and locations inside 32 bits, e.g. ARGB 32-bit (8-8-8-8), RGB 16-bit(5-6-5), ARGB 16-bit(4-4-4-4).
Input image size (combined mode)	48x48 to 8176x8176(66.8 Mpixels) Step size 16 pixels
Input image size (stand-alone mode)	Width from 48 to 8176 Height from 48 to 8176 Maximum size limited to 66.8 Mpixels Step size 16 pixels
Output image size	16x16 to 2560x4088 (when there is no scale, it can support 4088x4088) Horizontal step size 8 Vertical step size 2
Image up-scaling <sup>①</sup>	Bicubic polynomial interpolation with a four-tap horizontal kernel and a two-tap vertical kernel Arbitrary, non-integer scaling ratio, separately for both dimensions. Maximum output width is 3x the input width(within the maximum output image size limit) Maximum output height is 3x the input height -2 pixels (within the maximum output image size limit) Maximum output height is 2.5x the input height - 2 pixels (within the maximum output image size limit) when running RealVideo, VP8 format in pipeline
Image down-scaling <sup>①</sup>	Proprietary averaging filter Arbitrary, non-integer scaling ratio separately for both dimensions Unlimited down-scaling ratio
YCbCr to RGB color conversion	BT.601-5 compliant BT.709 compliant User definable conversion coefficient
Dithering	allego dithering for 4,5 and 6 bit RGB channel precision

RGB image contrast adjustment	Segmented linear
RGB image brightness adjustment	Linear
RGB image color saturation adjustment	Linear
De-blocking filter for MPEG-4 simple profile /H.263 /Sorenson	Using a modified H.264 in-loop filter as a post-processing filter. Filtering has to be performed in combined mode
Image cropping / digital zoom	User definable start position, height and width. Can be used with scaling to perform digital zoom. Usable only for JPEG or stand-alone mode.
Picture in picture	Output image can be written to any location inside video memory. Up to 2560x4088 sized displays supported. (when there is no scale, the size can up to 4088x4088)
Image rotation	Rotation 90,180, or 270 degrees Horizontal flip Vertical flip

①It is not allowed to perform horizontal up-scaling and vertical down-scaling (or vice versa) at the same. If needed, this kind of operation can be performed in two phases.

The PP has two modes: standalone mode and pipe-line mode. In standalone mode, picture processing is performed to any external source. The processing is done independently and asynchronously from the video decoder. The dataflow block gram is as Fig. 25-11 shows.

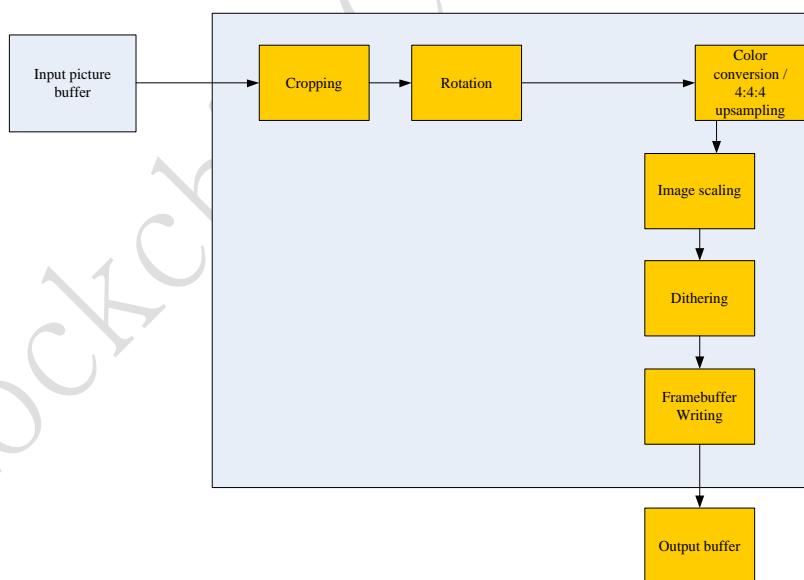


Fig. 25-11 Post-process standalone dataflow

In pipe-line mode, the post-processor works together with the multi-format decoder. The PP will take its input directly from the decoder. The post-processor doesn't have cropping function in pipe-line mode other than combined with jpeg decoder. The dataflow is as Fig. 25-12 show. In the pipe-line mode, most decoder will also put the data to the decoder out buffer other than JPEG decoder. So, JPEG decoder with pipe-line mode will save bus bandwidth when it crops the input picture to a smaller picture.

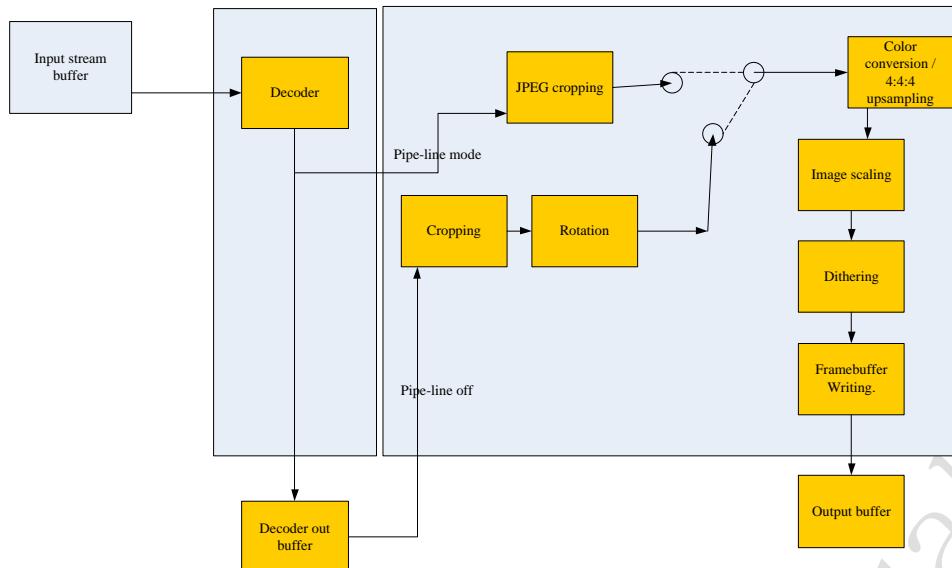


Fig. 25-12 Post-process Pipe-line Mode Dataflow

The post-processor has some restrictions in the input and output picture size. Table 25-13 presents the divisibility requirements for all the post-processor parameters.

Table 25-13 Requirements for post-processor

<b>Output format parameters</b>	<b>YCbCr 4:2:0</b>	<b>YCbCr 4:2:2</b>	<b>RGB16bpp</b>	<b>RGB32bpp</b>
Input picture width and height	16	16	16	16
Cropped picture width and height	8	8	8	8
Cropping start coordinates(x,y)	16	16	16	16
Output picture width	8	8	8	8
Output picture height	2	2	2	2
Masks width and origin X	8	4	4	2
Masks width and origin Y	2	1	1	1
Frame buffer width and origin X	8	4	4	2
Frame buffer height and origin Y	2	1	1	1

## 25.8 Image Pre-processor

Pre-processor is pipelined with the encoder and it can be used only with the encoder. Pre-processor features are presented in Table 25-14.

Table 25-14 Post-processor features

<b>Feature</b>	<b>Encoder support</b>
RGB to YCbCr 4:2:0 color space	BT.601, BT.709 or user defined

conversion	coefficients conversion for RGB: <ul style="list-style-type: none"> <li>● RGB444 and BGR444</li> <li>● RGB555 and BRG555</li> <li>● RGB565 and BGR565</li> <li>● RGB888 and BRG888</li> <li>● RGB101010 and BRG101010</li> </ul>
YCbCr 4:2:2 to YCbCr 4:2:0 color space conversion	YCbCr formats: <ul style="list-style-type: none"> <li>● YCbCr 4:2:0 planar</li> <li>● YCbCr 4:2:0 semi-planar</li> <li>● YCbYCr 4:2:2</li> <li>● CbYCrY 4:2:2 interleaved</li> </ul>
Cropping	Video – from 8192x8192 to any supported encoding size
Rotation	90 or 270 degrees

## 25.9 H.264 Encoder

The H.264 features supported by encoder are as shown in Table 25-15 .

Table 25-15 Video encoder H.264 feature

Feature	Encoder support
Input data format	<ul style="list-style-type: none"> <li>● YCbCr formats: YCbCr 4:2:0 planar YCbCr 4:2:0 semi-planar YCbYCr 4:2:2<sup>①</sup> CbYCrY 4:2:2 Interleaved<sup>①</sup></li> <li>● RGB formats:<sup>①</sup> RGB444 to BGR444 RGB555 to BGR555 RGB565 to BGR565 RGB888 to BRG888 RGB101010 and BRG 101010</li> </ul>
Output data format	H.264: Byte unit stream NAL unit stream
Supported image size	96x96 to 1920x1080(Full HD) Step size 4 pixels
Maximum frame rate	30 fps at 1920 x1080
Bit rate	Maximum 20Mbps Minimum 10kbps

①internally encoder handles image only in 4:2:0 format

Figure Fig. 25-13 illustrates the encoder data flow in H.264 encoding mode. The numbers present the following transactions:

1. Memory-mapped register writes and reads
2. Input image read
3. Reference image write
4. Reference image read
5. NAL sizes write from HW
6. NAL sizes read to SW
7. Output byte or NAL unit stream write from HW

## 8. Output byte or NAL unit stream headers write from SW

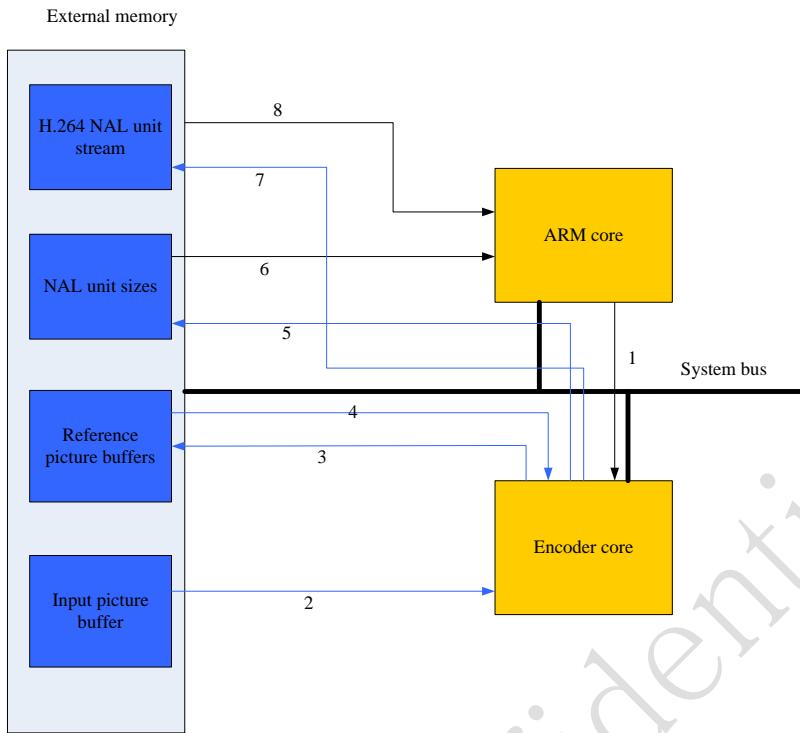


Fig. 25-13 Video Encoder Dataflow

The encoder software starts encoding the first picture by initializing hardware and writing the stream headers. After HW has encoded the image, SW calculates new quantization values for HW, and initializes HW again.

## 25.10 JPEG Encoder

The JPEG features supported by the encoder are as shown in Table 25-16.

Table 25-16 JPGE features

Feature	Encoder support
Input data format	<ul style="list-style-type: none"> <li>● YCbCr formats: YCbCr 4:2:0 planar YCbCr 4:2:0 semi-planar YCbCr 4:2:2<sup>①</sup> CbYCrY 4:2:2 Interleaved <sup>①</sup></li> <li>● RGB formats: RGB444 and BGR444 RGB555 and BGR555 RGB565 and BGR565 RGB888 and BRG888 RGB101010 and BRG101010</li> </ul>
Output data format	JFIF ifle format 1.02 Non-progressive JPEG
Supported image size	96x32 to 8192x8192(64 million pixels) Step size 4 pixels
Maximum data rate	Up to 90 million pixels per second
Thumbnail insertion	RGB 8-bits, RGB 24-bits and JPEG compressed thumbnails supported

<sup>①</sup>internally encoder handles images only in 4:2:0 format

## 25.11 MMU

The MMU divides memory into 4KB pages, where each page can be individually configured. For each page the following parameters are specified:

- Address translation of virtual memory, this enables the processor to work using address that differ from the physical address in the memory system.
- The permitted types of accesses to that page. Each page can permit read, write, both, or none.

The MMU use 2-level page table structure:

1. The first level, the page directory consists of 1024 directory table entries(DTEs), each pointing to a page table.
2. The second level, the page table consists of 1024 page table entries(PTEs), each pointing to a page in memory.

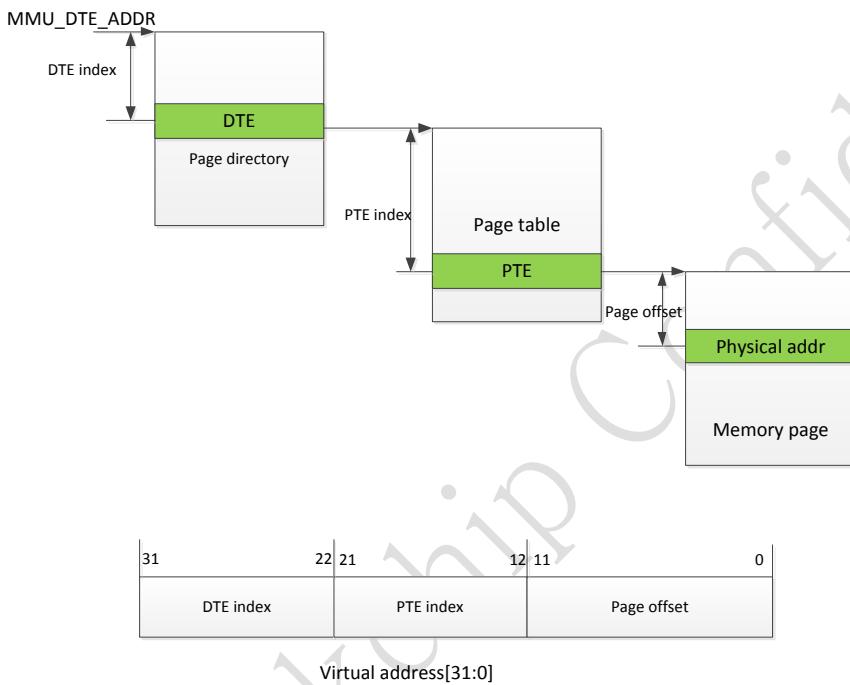


Fig. 25-14 structure of two-level page table

## 25.12 Register Description

This section describes the control/status registers of the design.

The VEPU base address is 0xff9a0000, the VDPU base address is 0xff9a0400.

The MMU base address is 0xff9a0800, the VDPU cache control base address is 0xff0a0c00.

Please refer to the document VDPU\_SWReg\_Map.pdf and VEPU\_SWReg\_Map.pdf.

The table below is the add new feature on the VDPU\_SWReg\_Map.pdf

### 25.12.1 VDPU new feature detail register description

#### SWREG15

Address: Operational Base + offset (0x03c)

bit	Attr	Reset Value	Description
19	RW	0x0	Sw_jpegroi_in_endian 0 = big endian (0-1-2-3 order) 1 = little endian (3-2-1-0 order)
18	RW	0x0	Sw_jpegroi_in_swap32 '0' = no swapping of 32 bit words '1' = 32bit data words are swapped (needed in 64 bit environment to achieve 7-6-5-4-3-2-1 byte order (also little endian should be enabled))
17:16	RW	0x0	Sw_roi_sample_size[1:0] ROI MB num sample each time 00: 1 01: 8 10:16 11: 8
15:12	RW	0x0	Sw_roi_distance[3:0] The distance between the sample MB and ROI start MB
11:10	RW	0x0	Sw_roi_out_sel[1:0] 00: output control 01: output picture 10: output control and picture 11: output control
9	RW	0x0	Sw_roi_decode 1'b0: build offset/dc table 1'b1: ROI decode
8	RW	0x0	Sw_roi_en 1'b0: normal jpeg decode mode 1'b1: jpeg roi mode

**SWREG35**

Address: Operational Base + offset (0x08c)

bit	Attr	Reset Value	Description
31:2	RW	0x0	Sw_jpegdcoff_base

**SWREG36**

Address: Operational Base + offset (0x090)

bit	Attr	Reset Value	Description
16:0	RW	0x0	Sw_jpegdcoff_len, the number of 64bit jpegdcoff, it can be used both when sw_roi_decode is 1'b0 or 1'b1

**SWREG57**

Address: Operational Base + offset (0x0E4)

bit	Attr	Reset Value	Description
14:8	R	0x0	Debug_service[6:0] Debug_service[6:0] = {service_wr[2:0], service_rd[3:0]} Now, it is only for paral axi bus version
7	RW	0x0	sw_cache_en when sw_cache_en = 1'b1 and sw_pref_sigchan = 1'b1, the prefetch cache is enable
6	RW	0x0	sw_pref_sigchan when it is set to 1'b1, the prefetch data is a single channel

5	RW	0x0	Sw_axird_sel: default is 1'b0 1'b0: auto sel encoder axi signals and decoder axi signals 1'b1: sel decoder axi signals(it only use to set bu_dec_e to 1'b0 in the middle of a frame)
4	RW	0x1	Sw_parallel_bus: when it is set 1'b1, the axi support read and write service parallel; when it is set 1'b0, the axi only support read and write serial
3	RW	0x0	sw_intra_dbl3t: in chroma dc intra prediction, when this bit is enable, there will 3 cycle enhance for every block
2	RW	0x0	Sw_intra_dblspeed: intra double speed enable
1	RW	0x0	Sw_inter_dblspeed: inter double speed enable
0	RW	0x0	Sw_stream_len_hi: the extension bit of sw_stream_len

**SWREG58**

Address: Operational Base + offset (0x0E8)

bit	Attr	Reset Value	Description
30	R	0x0	Mvst_mv_req signal value
29	R	0x0	Debug_rlc_req: scst_rlc_req signal value
28	R	0x0	Debug_res_y_req: prtr_res_y_req signal value
27	R	0x0	Debug_res_c_req: prtr_res_c_req signal value
26	R	0x0	Debug_strm_da_e: strm_da_e signal value
25	R	0x0	Debug_framerdy: dfbu_framerdy signal value
24	R	0x0	Debug_filter_req: dfbu_req_e signal value
23	R	0x0	Debug_referreq0: prbu_referreq0 signal value
22	R	0x0	Debug_referreq1: Prbu_referreq1 signal value
20:0	R	0x0	Debug_dec_mb_counter: HW internal MB counter value , it is now only reset by dec_e pulse

**SWREG98**

Address: Operational Base + offset (0x18C)

bit	Attr	Reset Value	Description
1	RW	0x0	Sw_pp_out_h_ext
0	RW	0x0	Sw_pp_out_w_ext

**SWREG101**

Address: Operational Base + offset (0x194)

bit	Attr	Reset Value	Description
0	W	0x0	Soft reset signals When write to 1'b1, it will reset VDPU, VEPU and pp when write to 1'b0, no use

The shared MMU register is as below

## 25.12.2 MMU Register Summary

Name	Offset	Size	Reset Value	Description
VCODEC_MMU_DTE_ADDR	0x0000	W	0x00000000	MMU current page Table address
VCODEC_MMU_STATUS	0x0004	W	0x00000000	MMU status register
VCODEC_MMU_COMMAND	0x0008	W	0x00000000	MMU command register
VCODEC_MMU_PAGE_FAULT_ADDR	0x000c	W	0x00000000	MMU logical address of last page fault
VCODEC_MMU_ZAP_ONE_LINE	0x0010	W	0x00000000	MMU Zap cache line register
VCODEC_MMU_INT_RAWSTAT	0x0014	W	0x00000000	MMU raw interrupt status register
VCODEC_MMU_INT_CLEAR	0x0018	W	0x00000000	MMU raw interrupt status register
VCODEC_MMU_INT_MASK	0x001c	W	0x00000000	MMU raw interrupt status register
VCODEC_MMU_INT_STATUS	0x0020	W	0x00000000	MMU raw interrupt status register
VCODEC_MMU_AUTO_GATING	0x0024	W	0x00000001	mmu auto gating

Notes: **Size** : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

## 25.12.3 MMU Detail Register Description

### VCODEC\_MMU\_DTE\_ADDR

Address: Operational Base + offset (0x0000)

MMU current page Table address

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	MMU_DTE_ADDR MMU_DTE_ADDR MMU current page Table address

### VCODEC\_MMU\_STATUS

Address: Operational Base + offset (0x0004)

MMU status register

Bit	Attr	Reset Value	Description
31:11	RO	0x0	reserved
10:6	RO	0x00	PAGEFAULT_BUS_ID page fault bus id Index of master responsible for last page fault
5	RO	0x0	PAGEFAULT_IS_WRITE page fault access The direction of access for last page fault: 0 = Read 1 = Write
4	RO	0x0	REPLAY_BUFFER_EMPTY replay buffer empty The MMU replay buffer is empty
3	RO	0x0	MMU_IDLE mmu idle The MMU is idle when accesses are being translated and there are no unfinished translated accesses.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RO	0x0	STAIL_ACTIVE stall active MMU stall mode currently enabled. The mode is enabled by command
1	RO	0x0	PAGE_FAULT_ACTIVE page fault active MMU page fault mode currently enabled . The mode is enabled by command.
0	RO	0x0	PAGING_ENABLED Paging is enabled Paging is enabled

**VCODEC\_MMU\_COMMAND**

Address: Operational Base + offset (0x0008)

MMU command register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2:0	WO	0x0	MMU_CMD mmu cmd MMU_CMD. This can be: 0: MMU_ENABLE_PAGING 1: MMU_DISABLE_PAGING 2: MMU_ENABLE_STALL 3: MMU_DISABLE_STALL 4: MMU_ZAP_CACHE 5: MMU_PAGE_FAULT_DONE 6: MMU_FORCE_RESET

**VCODEC\_MMU\_PAGE\_FAULT\_ADDR**

Address: Operational Base + offset (0x000c)

MMU logical address of last page fault

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	PAGE_FAULT_ADDR page fault addr address of last page fault

**VCODEC\_MMU\_ZAP\_ONE\_LINE**

Address: Operational Base + offset (0x0010)

MMU Zap cache line register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	WO	0x00000000	MMU_ZAP_ONE_LINE zap one line address to be invalidated from the page table cache

**VCODEC\_MMU\_INT\_RAWSTAT**

Address: Operational Base + offset (0x0014)

MMU raw interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	READ_BUS_ERROR read bus error read bus error status
0	RW	0x0	PAGE_FAULT page fault page fault status

**VCODEC\_MMU\_INT\_CLEAR**

Address: Operational Base + offset (0x0018)

MMU raw interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	WO	0x0	READ_BUS_ERROR read bus error write 1 to clear read bus error
0	WO	0x0	PAGE_FAULT page fault clear write 1 to page fault clear

**VCODEC\_MMU\_INT\_MASK**

Address: Operational Base + offset (0x001c)

MMU raw interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	READ_BUS_ERROR read bus error enable the read bus interrupt source when this bit is set to 1'b1
0	RW	0x0	PAGE_FAULT page fault mask enable the page fault interrupt source when this bit is set to 1'b1

**VCODEC\_MMU\_INT\_STATUS**

Address: Operational Base + offset (0x0020)

MMU raw interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RO	0x0	READ_BUS_ERROR read bus error read bus error status

Bit	Attr	Reset Value	Description
0	RO	0x0	PAGE_FAULT page fault page fault status

**VCODEC\_MMU\_AUTO\_GATING**

Address: Operational Base + offset (0x0024)

mmu auto gating

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x1	mmu_auto_clkgating mmu_auto_clkgating when it is 1'b1, the mmu will auto gating it self

**25.12.4 VDPU pref\_cache Register Summary**

Name	Offset	Size	Reset Value	Description
pref_cache_VERSION	0x0000	W	0xcac20101	VERSION register
pref_cache_SIZE	0x0004	W	0x06170206	L2 cache SIZE
pref_cache_STATUS	0x0008	W	0x00000000	Status register
pref_cache_COMMAND	0x0010	W	0x00000000	Command setting register
pref_cache_CLEAR_PAGE	0x0014	W	0x00000000	clear page register
pref_cache_MAX_READS	0x0018	W	0x0000001c	maximum read register
pref_cache_PERFCNT_SRC0	0x0020	W	0x00000000	performance counter 0 source register
pref_cache_PERFCNT_VAL0	0x0024	W	0x00000000	performance counter 0 value register
pref_cache_PERFCNT_SRC1	0x0028	W	0x00000000	performance counter 0 source register
pref_cache_PERFCNT_VAL1	0x002c	W	0x00000000	performance counter 1 value register

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

**25.12.5 VDPU pref\_cache Detail Register Description****pref\_cache\_VERSION**

Address: Operational Base + offset (0x0000)

VERSION register

Bit	Attr	Reset Value	Description
31:16	RO	0xcac2	PRODUCT_ID
15:8	RO	0x01	VERSION_MAJOR
7:0	RO	0x01	VERSION_MINOR

**pref\_cache\_SIZE**

Address: Operational Base + offset (0x0004)

L2 cache SIZE

Bit	Attr	Reset Value	Description

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x06	External_bus_width Log2 external bus width in bits
23:16	RO	0x17	CACHE_SIZE Log2 cache size in bytes
15:8	RO	0x02	ASSOCIATIVITY Log2 associativity
7:0	RO	0x06	LINE_SIZE Log2 line size in bytes

**pref\_cache\_STATUS**

Address: Operational Base + offset (0x0008)

Status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	DATA_BUSY set when the cache is busy handling data
0	RW	0x0	CMD_BUSY set when the cache is busy handling commands

**pref\_cache\_COMMAND**

Address: Operational Base + offset (0x0010)

Command setting register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:4	RW	0x0	sw_addrb_sel 2'b00: to sel b[14:6] 2'b01: to sel b[15:9], b[7:6] 2'b10: to sel b[16:10], b[7:6] 2'b11: to sel b[17:11], b[7:6]
3	RO	0x0	reserved
2:0	RW	0x0	COMMAND The possible command is 1 = Clear entire cache

**pref\_cache\_CLEAR\_PAGE**

Address: Operational Base + offset (0x0014)

clear page register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	CLEAR_PAGE writing an address, invalidates all lines in that page from the cache

**pref\_cache\_MAX\_READS**

Address: Operational Base + offset (0x0018)

maximum read register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RW	0x1c	MAX_READS Limit the number of outstanding read transactions to this amount

**pref\_cache\_PERFCNT\_SRC0**

Address: Operational Base + offset (0x0020)

Performance counter 0 source register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x0	PERFCNT_SRC0 This register holds all the possible source values for Performance Counter 0 0: total clock cycles 1: active clock cycles 2: read transactions, master 3: word reads, master 4: read transactions, slave 5: word reads, slave 6: read hit, slave 7: read misses, slave 8: read invalidates, slave 9: cacheable read transactions, slave 10: bad hit number, slave

**pref\_cache\_PERFCNT\_VAL0**

Address: Operational Base + offset (0x0024)

performance counter 0 value register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	PERFCNT_VAL0 Performance counter 0 value

**pref\_cache\_PERFCNT\_SRC1**

Address: Operational Base + offset (0x0028)

performance counter 0 source register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved

Bit	Attr	Reset Value	Description
3:0	RW	0x0	<p>PERFCNT_SRC1</p> <p>This register holds all the possible source values for Performance Counter 1</p> <p>0: total clock cycles      1: active clock cycles      2: read transactions, master      3: word reads, master      4: read transactions, slave      5: word reads, slave      6: read hit, slave      7: read misses, slave      8: read invalidates, slave      9: cacheable read transactions, slave      10: bad hit number, slave</p>

**pref\_cache\_PERFCNT\_VAL1**

Address: Operational Base + offset (0x002c)

Performance counter 1 value register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	<p>PERFCNT_VAL1</p> <p>Performance counter 1 value</p>

**25.13 Timing Diagram**

Fig. 25-15 illustrates the internal clock structure of VCODEC. VCODEC has two clocks input, which are aclk\_vcodec and hclk\_vcodec. They can be different frequency, but they should be in the same clock domains.

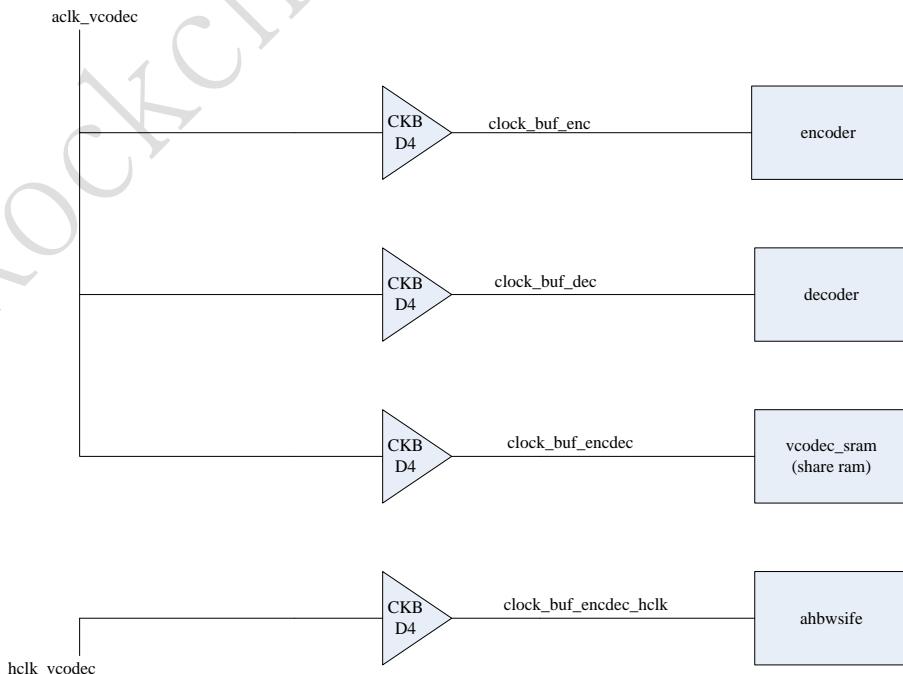


Fig. 25-15 VCODEC clock structure

The Fig. 25-16 shows the aclk\_vcodec and hclk\_vcodec architecture in the CRU module. Most signals are from the CRU register CRU\_CLKSEL32\_CON, while only one signal is from the GRF register GRF\_SOC\_CON0.

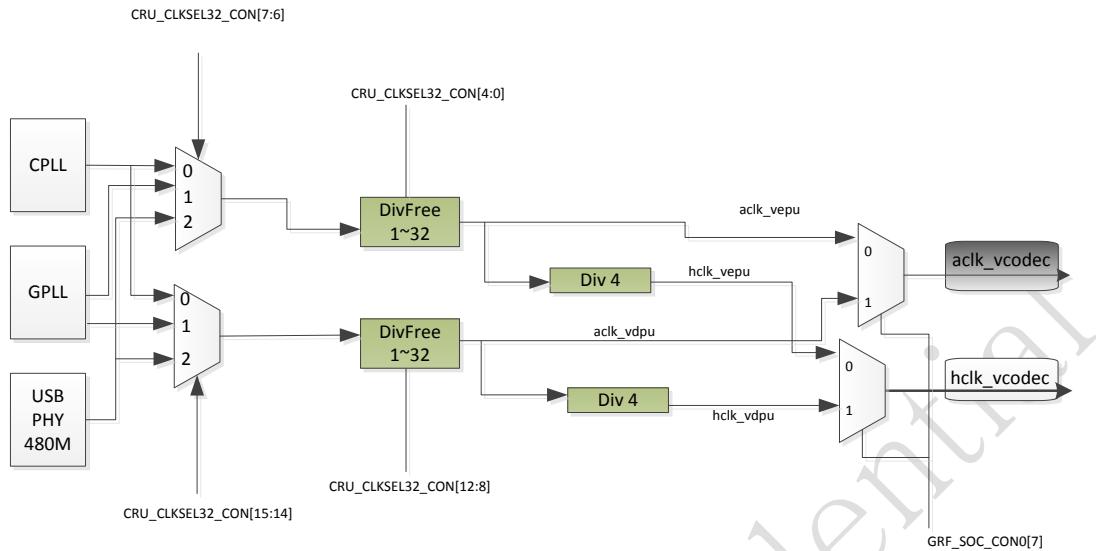


Fig. 25-16 Aclk\_vcodec and Hclk\_vcodec Architecture

When the encoder is working, the maximum frequency of aclk\_vcodec is 400MHz@worst case. The maximum frequency of aclk\_vcodec is also 400MHz@worst case when the decoder is working.

## 25.14 Interface Description

VCODEC supports writing and reading its internal registers through AHB bus and it just supports single 32bits read and write.

VCODEC reads the input data and write the output data through AXI bus. The VCODEC AXI master supports up to 32 outstanding bursts (in read) in most cases, while 24 bursts are the maximum in write. There are multiple bursts issued as outstanding bursts, they have the same ID. When the ID would change, the previous ID transactions are first completed. So the VCODEC AXI master doesn't support out of order.

VCODEC has three interrupt output signals vdpdu\_intr and vepu\_intr and irq\_mmu, which are high valid.

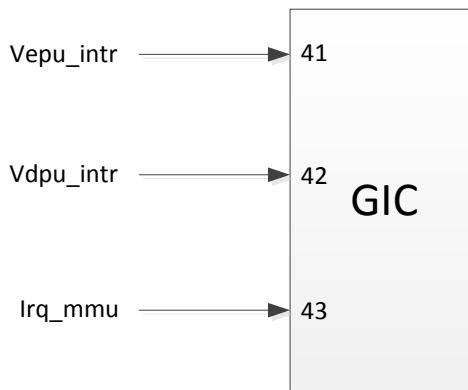


Fig. 25-17 The interrupt interface of vcodec

## 25.15 Application Notes

- In decoder other than JPEG decoder, the input stream buffer should at least contain a slice or a frame data, otherwise the decoder will produce an interrupt and show error and then reset itself.
- In encoder, we can configure the registers to control the input picture data format (such as endian and swap), but some input data format are fixed, such as cabac\_table data.
- The register VEPU\_SWREG64~95 are JPEG quantization registers. They are write only registers. When you want to write these registers, you should first set VEPU\_SWREG14[0] to 1'b0 and VEPU\_SWREG14[2:1] to 2'b10( select JPEG mode).
- The decoder can support ref buffer mode or cacheable mode, but they can't be both enabled. We can config the swreg57[6],swreg57[7] to enable cache and config the swreg51 to control the ref buffer.

## **Chapter 26 HEVC**

Will update soon!

Rockchip Confidential

## Chapter 27 Visual Output Processor (VOP)

### 27.1 Overview

VOP is the display interface from memory frame buffer to display device (LCD panel, LVDS, MIPI, eDP, HDMI and TV set). VOP is connected to an AHB bus through an AHB slave and AXI bus through an AXI master. The register setting is configured through the AHB slave interface and the display frame data is read through the AXI master interface. Furthermore, there is a data path between IEP and VOP, which can provide frame data from IEP to VOP.

#### 27.1.1 Features

- Display interface
  - Parallel RGB LCD Interface
    - ◆ RGB101010,RGB888,RGB666,RGB565
  - Serial RGB LCD Interface
    - ◆ 2x12-bit,3x8-bit(RGB delta supported),3x8-bit+dummy
  - Parallel MCU LCD Interface
    - ◆ 24-bit(RGB888),18-bit(RGB666),16-bit(RGB565)
    - ◆ hold/auto/bypass mode
  - Serial MCU LCD Interface
    - ◆ 2x12-bit, 3x8-bit with hold mode
  - TV Interface
    - ◆ ITU-R BT.656(8-bit, 480i/576i/1080i)
    - ◆ 3 output mode: valid data in lower 8bit, middle 8bit and higher 8bit
    - ◆ TV encoder
  - Support SDR(single data rate) interface
  - Support DDR(dual data rate) interface for LVDS/PARALLEL RGB
    - ◆ parallel RGB and 2x12-bit serial RGB
    - ◆ Single or dual clock out
  - Max output resolution
    - ◆ VOP\_BIG: 3840x2160
    - ◆ VOP\_LITTLE: 2560x1600
  - Scaning timing 8192x4096
  - Support configurable polarity of DCLK/HSYNC/VSYNC/DEN
  - 4 groups of scanning output for PARALLEL RGB,LVDS,LVDS,HDMI,eDP
  - MIPI control
    - ◆ MIPI dual channel,overlay scan(overlapped pixels=2~16 pix)
    - ◆ MIPI flow control(edpihalt)
    - ◆ MIPI DCS command mode
- Display process
  - CABC
  - BCSH,8bit
    - ◆ Brightness,Contrast,Saturation,Hue adjustment
    - ◆ YUV-10bit, RGB-10bit
  - Dither down
    - ◆ pre dither down for RGB-10bit to RGB-8bit
    - ◆ allegro for RGB565 and RGB666
    - ◆ FRC with configurable pattern for RGB666
  - Gamma
    - ◆ LUT(lookup table) for R/G/B respectively
    - ◆ 8bit/10bit RGB look up table
    - ◆ gamma after dither

- Support display data swap
  - ◆ BG swap, RB swap, RG swap, dummy swap, delta swap
- Support three YUV2RGB transition modes:
  - ◆ 8bit-YUV: rec601-mpeg/rec601-jpeg/rec709
  - ◆ 10bit-YUV: BT2020
- blank display
- black display
- standby mode
- auto dynamic power control
- X-MIRROR,Y-MIRROR for win0/win1/win2/win3/hwc?
- scale down for TV overscan
  - ◆ after overlay
  - ◆ arbitrary non-integer scaling ratio
  - ◆ horizontal scale down using bilinear, 0.5~1.0
  - ◆ vertical scale down using bilinear, 0.5~1.0
- Layer process
  - Background layer
    - ◆ programmable 24-bit color
  - Win0/Win1 layer
    - ◆ Support data format
      - ✧ RGB888, ARGB888, RGB565,
      - ✧ YCbCr420SP, YCbCr422SP, YCbCr444SP
      - ✧ YUV-8bit,YUV-10bit
    - ◆ YUV clip
      - ✧ Y-10bit:64~940;UV-10bit: 64~960
      - ✧ Y-8bit: 16~235;UV-8bit: 16~240
    - ◆ Support max input resolution 4096x8192
    - ◆ Support max output resolution 3840x2160
    - ◆ Support virtual display
    - ◆ Support 1/8 to 8 scaling-down and scaling-up engine
      - ✧ scale up using bicubic and bilinear
        - Arbitrary non-integer scaling ratio
        - 4 bicubic table for scale up using precise,spline,catrom,Mitchell
      - ✧ scale down using bilinear and average
        - Arbitrary non-integer scaling ratio
      - ✧ per-pix alpha + scale
    - ◆ Support data swap
      - ✧ RGB/BPP: alpha\_swap,rb\_swap
      - ✧ YUV: mid\_swap,uv\_swap
    - ◆ transparency color key,prior to alpha blending and fading
    - ◆ Support fading,prior to alpha blending
    - ◆ Support alpha blending
    - ◆ Support interlace and de-flicker for interlace output
    - ◆ Support IEP direct path input
  - Win2/Win3 layer
    - ◆ Support data format
      - ✧ RGB888, ARGB888, RGB565
      - ✧ 1BPP,2BPP,4BPP,8BPP
      - ✧ little endian and big endian for BPP
      - ✧ BYPASS and LUT mode(25bit LUT, 1bit AA+8bit-RGB) for BPP
    - ◆ 4 display regions
      - ✧ only one region at one scanning line
    - ◆ Support data swap
      - ✧ RGB/BPP:rb\_swap,alpha\_swap
    - ◆ Support transparency color key,prior to alpha blending and fading
    - ◆ Support fading,prior to alpha blending
    - ◆ Support alpha blending

- ◆ Support interlace read and interlace output
- ◆ Support IEP direct path input
- Hardware Cursor layer(HWC for short)
  - ◆ Support data format
    - ✧ RGB888, ARGB888, RGB565
    - ✧ 1BPP,2BPP,4BPP,8BPP
    - ✧ little endian and big endian for BPP
    - ✧ BYPASS and LUT mode(25bit LUT, 1bit AA+8bit-RGB)for BPP
  - ◆ Support four hwc size: 32x32,64x64,96x96,128x128
  - ◆ Support 2 color modes: normal and reversed color
  - ◆ Support fading,prior to alpha blending
  - ◆ Support alpha blending
  - ◆ Support displaying out of panel,right or bottom
  - ◆ Support NORMAL color and REVERSE color mode
  - ◆ Support interlace read and interlace output
- Overlay
  - ◆ Support 6 layers,background/win0/win1/win2/win3/hwc
  - ◆ Win0/Win1/Win2/Win3 overlay position exchangeable
  - ◆ Alpha blending
    - ✧ Support 12 alpha blending modes
    - ✧ Support pre-multiplied alpha
    - ✧ Support global alpha and per\_pix alpha
    - ✧ Support 256 level alpha
    - ✧ layer1/layer2/layer3/hwc support alpha
- Bus interface
  - Support AMBA 2.0 AHB slave interface for accessing internal registers and LUT memories, 32bit data bus width
  - Support AMBA 3.0 AXI master read interface for loading frame data
    - ◆ 128bit data bus width
  - Support MMU
  - Support two transfer modes
    - ◆ auto outstanding transfer
    - ◆ configuruable outstanding transfer(gather transfer)
  - DMA line mode for YUV
  - Support QoS request for higher bus priority for win2/win3
  - Support NOC hurry for higher bus prioirty for win0/win1
  - Support DMA stop mode
  - max read outstanding number
    - ◆ 32 when MMU disable
    - ◆ 31 when MMU enable
- Interrupt
  - One combined interrupt
    - ◆ high active
    - ◆ raw status
    - ◆ combinational with 12 interrupt sources
      - ✧ frame start interrupt
      - ✧ line flag interrupt
      - ✧ bus error interrupt
      - ✧ win0 empty interrupt
      - ✧ win1 empty interrupt
      - ✧ win2 empty interrupt
      - ✧ win3 empty interrupt
      - ✧ hwc empty interrupt
      - ✧ post empty interrupt
      - ✧ pwm gen interrupt

✧ irq\_mmu

## 27.2 Block Diagram

The architecture is shown in the following figure.

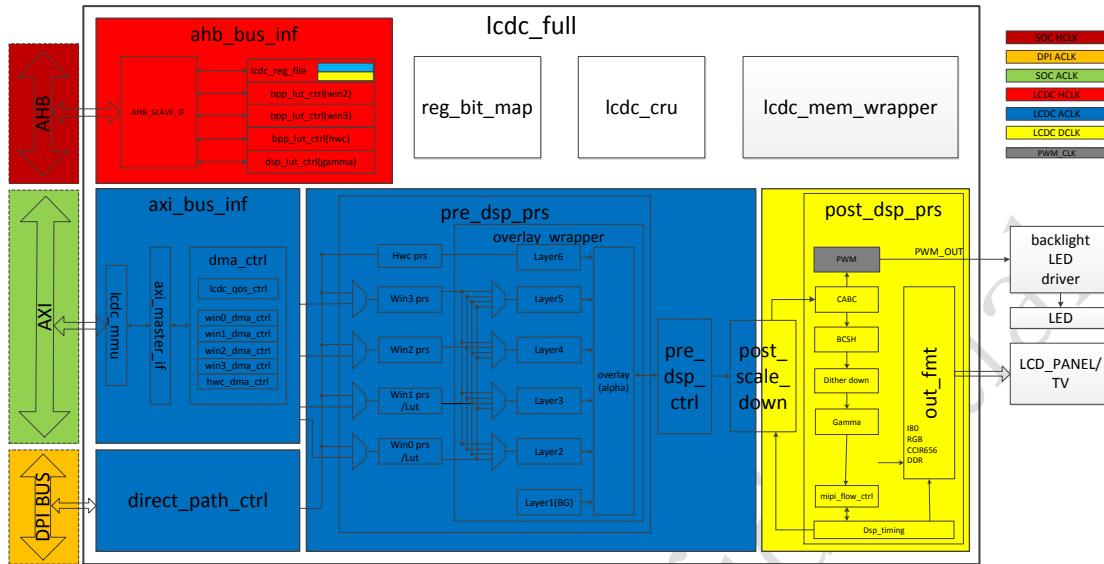


Fig. 27-1 VOP Block Diagram

## 27.3 Function Description

### 27.3.1 Pixel format

#### 1.RGB

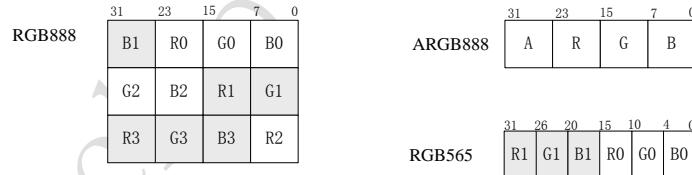


Fig. 27-2 RGB data format

#### 2.YCbCr/YUV(8bit/10bit)

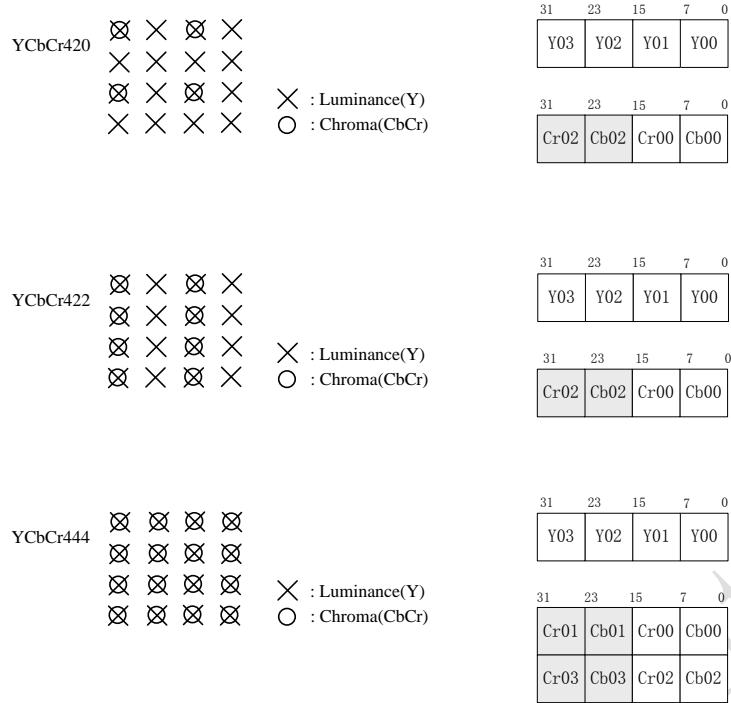


Fig. 27-3 YUV data format

YUV just support SP

YUV-8bit 32bit align

YUV-10bit 128bit align

### 3.BPP

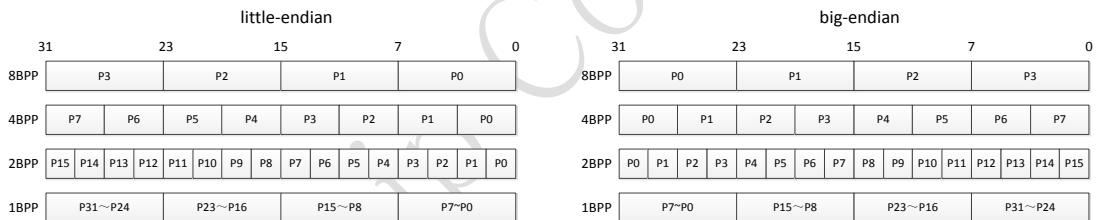


Fig. 27-4 BPP little/big endian data format

### 27.3.2 Pixel Data Path

There are two data input path for VOP to get display layers' pixel data. One is internal DMA; the other is direction path interface.

#### 1.Internal DMA

Internal DMA can fetch the pixel data through AXI bus from system memory (DDR) for all the display layers. Data fetching is driven by display output requirement.

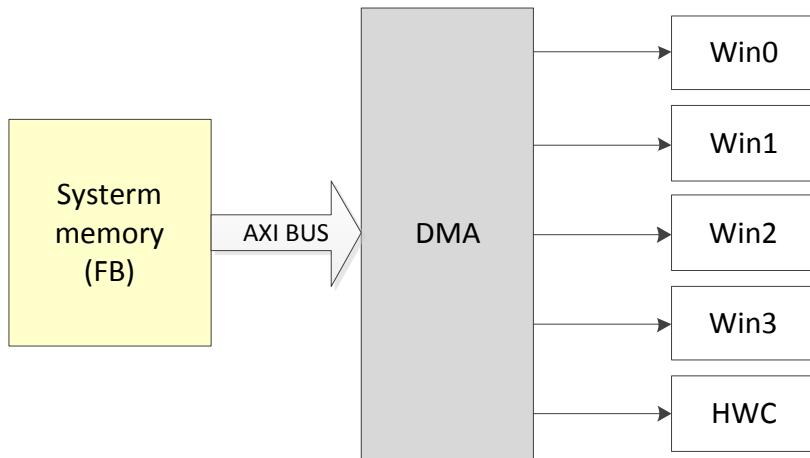


Fig. 27-5 LCD Internal DMA

## 2.Direct Path Interface

Direct path interface (DPI) is used for direct image display of external image processing IP. There is a local bus between VOP and external image processing IP for the data transfer.

DPI is connected to WIN0/WIN1/WIN2/WIN3 but can only be configured for One layer use (Win0 or Win1 or Win2 or Win3) in each frame.

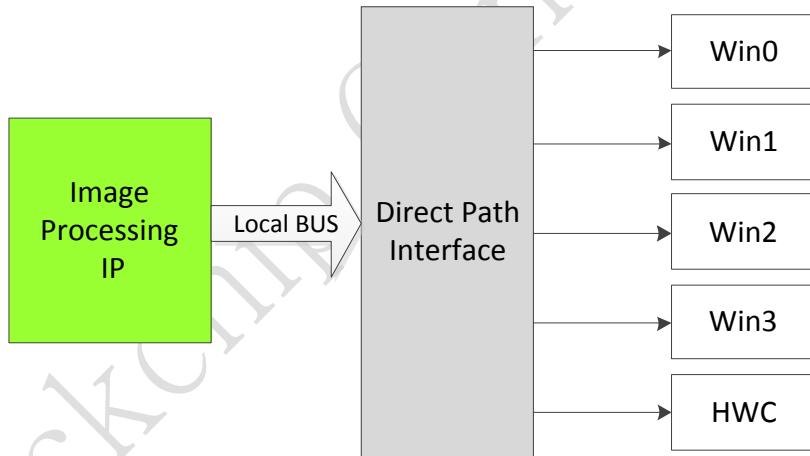


Fig. 27-6 VOP Direct Path Interface

### 27.3.3 Win Scaling

The scaling operation is the image resizing process by scaling-up or scaling-down the source image from active window size to display window size for displaying on LCD panel or TV set. Horizontal scaling and vertical scaling are realized independently.

#### 1.Scaling factor

Pseudo Code:

```

void calc_win_scl_factor(LCDC_WIN_PARAMETERS *p_win_para)
{
    u16 srcW;
    u16 srcH;
  
```

```

u16 dstW;
u16 dstH;
u8  is_3d_mix;
u16 yrgb_srcW;
u16 yrgb_srcH;
u16 yrgb_dstW;
u16 yrgb_dstH;
u32 yrgb_vScaleDnMult;
u32 yrgb_xscl_factor;
u32 yrgb_yscl_factor;
u8  yrgb_vsd_bil_gt2;
u8  yrgb_vsd_bil_gt4;
u8  yrgb_vsd_bil_extra;

u16 cbcr_srcW;
u16 cbcr_srcH;
u16 cbcr_dstW;
u16 cbcr_dstH;
u32 cbcr_vScaleDnMult;
u32 cbcr_xscl_factor;
u32 cbcr_yscl_factor;
u8  cbcr_vsd_bil_gt2;
u8  cbcr_vsd_bil_gt4;
u8  cbcr_vsd_bil_extra;

//-----
//width and height, 3D enable
is_3d_mix = (p_win_para->win_3d_en == ENABLE) && ((p_win_para->win_3d_mode == MIX_R_GB)
|| (p_win_para->win_3d_mode == MIX_G_RB) || (p_win_para->win_3d_mode == MIX_B_RG));

srcW  = p_win_para->win_act_width;
if((p_win_para->win_3d_en == ENABLE)&&(p_win_para->win_3d_mode ==
INTERLEAVE_HORIZONTAL)) {
    dstW  = p_win_para->dsp_win_width >> 1;
} else {
    dstW  = p_win_para->dsp_win_width;
}
srcH = p_win_para->win_act_height;

if((p_win_para->win_3d_en == ENABLE)&&(p_win_para->win_3d_mode == INTERLEAVE_VERTICAL))
{
    dstH = p_win_para->dsp_win_height >> 1;
} else {
    dstH = p_win_para->dsp_win_height;
}

dstH = p_win_para->dsp_win_height;

if(p_win_para->win_3d_en == ENABLE) {
    if(p_win_para->win_3d_mode == INTERLEAVE_HORIZONTAL) {
        dstW = dstW/2;
    }
    else if(p_win_para->win_3d_mode == INTERLEAVE_VERTICAL) {
        dstH = dstH/2;
    }
}

//-----
//SCALE MODE(YGRB)
yrgb_srcW = srcW;
yrgb_dstW = dstW;
yrgb_srcH = srcH;
yrgb_dstH = dstH;

printf("[hxx_dbg] yrgb_srcW=%d; yrgb_dstW=%d; yrgb_srcH=%d;
yrgb_dstH=%d;\n",yrgb_srcW,yrgb_dstW,yrgb_srcH,yrgb_dstH);

```

```

if (yrgb_srcW < yrgb_dstW) {
    p_win_para->yrgb_hor_scl_mode = SCALE_UP;
} else if (yrgb_srcW > yrgb_dstW) {
    p_win_para->yrgb_hor_scl_mode = SCALE_DOWN;
} else {
    p_win_para->yrgb_hor_scl_mode = SCALE_NONE;
}

if (yrgb_srcH < yrgb_dstH) {
    p_win_para->yrgb_ver_scl_mode = SCALE_UP;
} else if (yrgb_srcH > yrgb_dstH) {
    p_win_para->yrgb_ver_scl_mode = SCALE_DOWN;
} else {
    p_win_para->yrgb_ver_scl_mode = SCALE_NONE;
}

//SCALE MODE(CBCR)
if(p_win_para->win_lcdc_format == LCDC_FMT_YUV422) {
    cbcr_srcW = srcW/2;
    cbcr_dstW = dstW;
    cbcr_srcH = srcH;
    cbcr_dstH = dstH;

    if (cbcr_srcW < cbcr_dstW) {
        p_win_para->cbr_hor_scl_mode = SCALE_UP;
    } else if (cbcr_srcW > cbcr_dstW) {
        p_win_para->cbr_hor_scl_mode = SCALE_DOWN;
    } else {
        p_win_para->cbr_hor_scl_mode = SCALE_NONE;
    }

    if (cbcr_srcH < cbcr_dstH) {
        p_win_para->cbr_ver_scl_mode = SCALE_UP;
    } else if (cbcr_srcH > cbcr_dstH) {
        p_win_para->cbr_ver_scl_mode = SCALE_DOWN;
    } else {
        p_win_para->cbr_ver_scl_mode = SCALE_NONE;
    }
}
else if(p_win_para->win_lcdc_format == LCDC_FMT_YUV420) {
    cbcr_srcW = srcW/2;
    cbcr_dstW = dstW;
    cbcr_srcH = srcH/2;
    cbcr_dstH = dstH;

    if (cbcr_srcW < cbcr_dstW) {
        p_win_para->cbr_hor_scl_mode = SCALE_UP;
    } else if (cbcr_srcW > cbcr_dstW) {
        p_win_para->cbr_hor_scl_mode = SCALE_DOWN;
    } else {
        p_win_para->cbr_hor_scl_mode = SCALE_NONE;
    }

    if (cbcr_srcH < cbcr_dstH) {
        p_win_para->cbr_ver_scl_mode = SCALE_UP;
    } else if (cbcr_srcH > cbcr_dstH) {
        p_win_para->cbr_ver_scl_mode = SCALE_DOWN;
    } else {
        p_win_para->cbr_ver_scl_mode = SCALE_NONE;
    }
}
else if(p_win_para->win_lcdc_format == LCDC_FMT_YUV444) {
    cbcr_srcW = srcW;
    cbcr_dstW = dstW;
    cbcr_srcH = srcH;
    cbcr_dstH = dstH;
}

```

```

if (cbcr_srcW < cbcr_dstW) {
    p_win_para->cbr_hor_scl_mode = SCALE_UP;
} else if (cbcr_srcW > cbcr_dstW) {
    p_win_para->cbr_hor_scl_mode = SCALE_DOWN;
} else {
    p_win_para->cbr_hor_scl_mode = SCALE_NONE;
}

if (cbcr_srcH < cbcr_dstH) {
    p_win_para->cbr_ver_scl_mode = SCALE_UP;
} else if (cbcr_srcH > cbcr_dstH) {
    p_win_para->cbr_ver_scl_mode = SCALE_DOWN;
} else {
    p_win_para->cbr_ver_scl_mode = SCALE_NONE;
}
} else {
    cbcr_srcW = 0;
    cbcr_dstW = 0;
    cbcr_srcH = 0;
    cbcr_dstH = 0;

    p_win_para->cbr_hor_scl_mode = SCALE_NONE;
    p_win_para->cbr_ver_scl_mode = SCALE_NONE;
}

printf("[hxx_dbg] cbcr_srcW=%d; cbcr_dstW=%d; cbcr_srcH=%d;
cbcr_dstH=%d;\n",cbcr_srcW,cbcr_dstW,cbcr_srcH,cbcr_dstH);

//-----
//SCALE ALGORITHM
if( (p_win_para->win_lcdc_format == LCDC_FMT_YUV422) || (p_win_para->win_lcdc_format ==
LCDC_FMT_YUV420) ) {
    if(p_win_para->cbr_hor_scl_mode == SCALE_DOWN) {
        if(cbcr_dstW > 3840) {
            printf("ERROR cbcr_dst_width exceeds 3840\n");
            exit (-1);
        } else if(cbcr_dstW > 2560) {
            p_win_para->win_lb_mode = LB_RGB_3840X2;
        } else if(cbcr_dstW > 1920) {
            if(p_win_para->yrgb_hor_scl_mode == SCALE_DOWN) {
                if(yrgb_dstW > 3840) {
                    printf("ERROR yrgb_dst_width exceeds 3840\n");
                    exit (-1);
                } else if(yrgb_dstW > 2560) {
                    p_win_para->win_lb_mode = LB_RGB_3840X2;
                } else if(yrgb_dstW > 1920) {
                    p_win_para->win_lb_mode = LB_RGB_2560X4;
                } else {
                    printf("ERROR never run here!yrgb_dstW<1920 ==> cbcr_dstW<1920");
                    exit (-1);
                }
            }
        } else if(cbcr_dstW > 1280) {
            p_win_para->win_lb_mode = LB_YUV_3840X5;
        } else {
            p_win_para->win_lb_mode = LB_YUV_2560X8;
        }
    } else { //SCALE_UP or SCALE_NONE
        if(cbcr_srcW > 3840) {
            printf("ERROR cbcr_act_width exceeds 3840\n");
            exit (-1);
        } else if(cbcr_srcW > 2560) {
            p_win_para->win_lb_mode = LB_RGB_3840X2;
        } else if(cbcr_srcW > 1920) {
            if(p_win_para->yrgb_hor_scl_mode == SCALE_DOWN) {
                if(yrgb_dstW > 3840) {

```

```

        printf("ERROR yrgb_dst_width exceeds 3840\n");
        exit (-1);
    } else if(yrgb_dstW > 2560) {
        p_win_para->win_lb_mode = LB_RGB_3840X2;
    } else if(yrgb_dstW > 1920) {
        p_win_para->win_lb_mode = LB_RGB_2560X4;
    } else {
        printf("ERROR never run here!yrgb_dstW<1920 ==> cbcr_dstW<1920 ==>
cbcr_srcW<=1920\n");
        exit (-1);
    }
}
} else if(cbcr_srcW > 1280) {
    p_win_para->win_lb_mode = LB_YUV_3840X5;
} else {
    p_win_para->win_lb_mode = LB_YUV_2560X8;
}
}
else {
    if(p_win_para->yrgb_hor_scl_mode == SCALE_DOWN) {
        if(yrgb_dstW > 3840) {
            printf("ERROR yrgb_DSP_width exceeds 3840\n");
            exit (-1);
        } else if(yrgb_dstW > 2560) {
            p_win_para->win_lb_mode = LB_RGB_3840X2;
        } else if(yrgb_dstW > 1920) {
            p_win_para->win_lb_mode = LB_RGB_2560X4;
        } else if(yrgb_dstW > 1280){
            p_win_para->win_lb_mode = LB_RGB_1920X5;
        } else {
            p_win_para->win_lb_mode = LB_RGB_1280X8;
        }
    } else { //SCALE_UP or SCALE_NONE
        if(yrgb_srcW > 3840) {
            printf("ERROR yrgb_ACT_width exceeds 3840\n");
            exit (-1);
        } else if(yrgb_srcW > 2560) {
            p_win_para->win_lb_mode = LB_RGB_3840X2;
        } else if(yrgb_srcW > 1920) {
            p_win_para->win_lb_mode = LB_RGB_2560X4;
        } else if(yrgb_srcW > 1280){
            p_win_para->win_lb_mode = LB_RGB_1920X5;
        } else {
            p_win_para->win_lb_mode = LB_RGB_1280X8;
        }
    }
}

printf("[hx_dbg] p_win_para->win_lb_mode = %d;\n",p_win_para->win_lb_mode);

//vsd/vsu scale ALGORITHM
switch(p_win_para->win_lb_mode) {
    case LB_YUV_3840X5:
        p_win_para->yrgb_vsu_mode = SCALE_UP_BIC ;
        //p_win_para->yrgb_vsd_mode = SCALE_DOWN_BIL; //not to specify
        p_win_para->cbr_vsu_mode = SCALE_UP_BIC ;
        //p_win_para->cbr_vsd_mode = SCALE_DOWN_BIL; //not to specify
        break;
    case LB_YUV_2560X8:
        p_win_para->yrgb_vsu_mode = SCALE_UP_BIC ;
        //p_win_para->yrgb_vsd_mode = SCALE_DOWN_BIL; //not to specify
        p_win_para->cbr_vsu_mode = SCALE_UP_BIC ;
        //p_win_para->cbr_vsd_mode = SCALE_DOWN_BIL; //not to specify
        break;
    case LB_RGB_3840X2:
        if(p_win_para->yrgb_ver_scl_mode != SCALE_NONE) {

```

```

        printf("ERROR : not allow yrzb ver scale\n");
        exit(-1);
    }

    if(p_win_para->cbr_ver_scl_mode != SCALE_NONE) {
        printf("ERROR : not allow cbcr ver scale\n");
        exit(-1);
    }

    //p_win_para->yrzb_vsu_mode = SCALE_UP_BIC ;
    //p_win_para->yrzb_vsd_mode = SCALE_DOWN_BIL;
    //p_win_para->cbr_vsu_mode = SCALE_UP_BIC ;
    //p_win_para->cbr_vsd_mode = SCALE_DOWN_BIL;
    break;
case LB_RGB_2560X4:
    p_win_para->yrzb_vsu_mode = SCALE_UP_BIL ;//<2
    //p_win_para->yrzb_vsd_mode = SCALE_DOWN_BIL; //<2 //not to specify
    p_win_para->cbr_vsu_mode = SCALE_UP_BIL ;//<2
    //p_win_para->cbr_vsd_mode = SCALE_DOWN_BIL; //<2 //not to specify
    break;
case LB_RGB_1920X5:
    p_win_para->yrzb_vsu_mode = SCALE_UP_BIC ;
    //p_win_para->yrzb_vsd_mode = SCALE_DOWN_BIL; //not to specify
    p_win_para->cbr_vsu_mode = SCALE_UP_BIC ;
    //p_win_para->cbr_vsd_mode = SCALE_DOWN_BIL; //not to specify
    break;
case LB_RGB_1280X8:
    p_win_para->yrzb_vsu_mode = SCALE_UP_BIC ;
    //p_win_para->yrzb_vsd_mode = SCALE_DOWN_BIL; //not to specify
    p_win_para->cbr_vsu_mode = SCALE_UP_BIC ;
    //p_win_para->cbr_vsd_mode = SCALE_DOWN_BIL; //not to specify
    break;
default :
    break;
}
//-----
//SCALE FACTOR
yrzb_vsd_bil_gt4 = 0;
yrzb_vsd_bil_gt2 = 0;
cbcr_vsd_bil_gt4 = 0;
cbcr_vsd_bil_gt2 = 0;

//(1.1)YRGB HOR SCALE FACTOR
switch(p_win_para->yrzb_hor_scl_mode) {
    case SCALE_NONE:
        yrzb_xscl_factor = (1<<SCALE_FACTOR_DEFAULT_FIXPOINT_SHIFT);
        break;

    case SCALE_UP :
        yrzb_xscl_factor = GET_SCALE_FACTOR_BIC(yrgb_srcW, yrgb_dstW);
        break;

    case SCALE_DOWN:
        switch(p_win_para->yrzb_hsd_mode)
        {
            case SCALE_DOWN_BIL:
                yrzb_xscl_factor = GET_SCALE_FACTOR_BILI_DN(yrgb_srcW, yrgb_dstW);
                break;
            case SCALE_DOWN_AVG:
                yrzb_xscl_factor = GET_SCALE_FACTOR_AVRG(yrgb_srcW, yrgb_dstW);
                break;
            default :
                break;
        } //p_win_para->yrzb_vsd_mode
        break;

    default :

```

```

        break;
} //p_win_para->yrgb_hor_scl_mode

//(1.2)YRGB VER SCALE FACTOR
switch(p_win_para->yrgb_ver_scl_mode)
{
    case SCALE_NONE:
        yrgb_yscl_factor = (1<<SCALE_FACTOR_DEFAULT_FIXPOINT_SHIFT);
        break;

    case SCALE_UP :
        switch(p_win_para->yrgb_vsu_mode)
        {
            case SCALE_UP_BIL:
                yrgb_yscl_factor = GET_SCALE_FACTOR_BILI_UP(yrgb_srcH, yrgb_dstH);
                break;
            case SCALE_UP_BIC:
                if(yrgb_srcH < 3) {
                    printf("[hxx_dbg] yrgb_srcH should be greater than 3 !!!\n");
                    exit (-1);
                }
                yrgb_yscl_factor = GET_SCALE_FACTOR_BIC(yrgb_srcH, yrgb_dstH);
                break;
            default :
                break;
        } //p_win_para->yrgb_vsu_mode
        break;

    case SCALE_DOWN:
        switch(p_win_para->yrgb_vsd_mode)
        {
            case SCALE_DOWN_BIL:
                yrgb_vScaleDnMult = getHardWareVSkipLines(yrgb_srcH, yrgb_dstH);
                yrgb_yscl_factor = GET_SCALE_FACTOR_BILI_DN_VSKIP(yrgb_srcH, yrgb_dstH,
yrgb_vScaleDnMult);
                //printf("[hxx_dbg] yrgb_vScaleDnMult=%d;
yrgb_yscl_factor=%#4x;\n",yrgb_vScaleDnMult,yrgb_yscl_factor);
                if(yrgb_vScaleDnMult == 4) {
                    yrgb_vsd_bil_gt4 = 1;
                    yrgb_vsd_bil_gt2 = 0;
                } else if(yrgb_vScaleDnMult == 2) {
                    yrgb_vsd_bil_gt4 = 0;
                    yrgb_vsd_bil_gt2 = 1;
                } else {
                    yrgb_vsd_bil_gt4 = 0;
                    yrgb_vsd_bil_gt2 = 0;
                }
                break;
            case SCALE_DOWN_AVG:
                yrgb_yscl_factor = GET_SCALE_FACTOR_AVRG(yrgb_srcH, yrgb_dstH);
                break;
            default :
                break;
        } //p_win_para->yrgb_vsd_mode
        break;

    default :
        break;
} //p_win_para->yrgb_hor_scl_mode

p_win_para->win_h_YRGB_factor = yrgb_xscl_factor;
p_win_para->win_v_YRGB_factor = yrgb_yscl_factor;
p_win_para->vsd_yrgb_gt4      = yrgb_vsd_bil_gt4;
p_win_para->vsd_yrgb_gt2      = yrgb_vsd_bil_gt2;

//(2.1)CBCR HOR SCALE FACTOR
switch(p_win_para->cbr_hor_scl_mode)

```

```

{
    case SCALE_NONE:
        cbcr_xscl_factor = (1<<SCALE_FACTOR_DEFAULT_FIXPOINT_SHIFT);
        break;

    case SCALE_UP :
        cbcr_xscl_factor = GET_SCALE_FACTOR_BIC(cbcr_srcW, cbcr_dstW);
        break;

    case SCALE_DOWN:
        switch(p_win_para->cbr_hsd_mode)
        {
            case SCALE_DOWN_BIL:
                cbcr_xscl_factor = GET_SCALE_FACTOR_BILI_DN(cbcr_srcW, cbcr_dstW);
                break;
            case SCALE_DOWN_AVG:
                cbcr_xscl_factor = GET_SCALE_FACTOR_AVRG(cbcr_srcW, cbcr_dstW);
                break;
            default :
                break;
        } //p_win_para->cbr_vsd_mode
        break;

    default :
        break;
} //p_win_para->cbr_hor_scl_mode

//(2.2)CBCR VER SCALE FACTOR
switch(p_win_para->cbr_ver_scl_mode)
{
    case SCALE_NONE:
        cbcr_yscl_factor = (1<<SCALE_FACTOR_DEFAULT_FIXPOINT_SHIFT);
        break;

    case SCALE_UP :
        switch(p_win_para->cbr_vsu_mode)
        {
            case SCALE_UP_BIL:
                cbcr_yscl_factor = GET_SCALE_FACTOR_BILI_UP(cbcr_srcH, cbcr_dstH);
                break;
            case SCALE_UP_BIC:
                if(cbcr_srcH < 3) {
                    printf("[hxx_dbg] cbcr_srcH should be greater than 3 !!!\n");
                    exit (-1);
                }
                cbcr_yscl_factor = GET_SCALE_FACTOR_BIC(cbcr_srcH, cbcr_dstH);
                break;
            default :
                break;
        } //p_win_para->cbr_vsu_mode
        break;

    case SCALE_DOWN:
        switch(p_win_para->cbr_vsd_mode)
        {
            case SCALE_DOWN_BIL:
                cbcr_vScaleDnMult = getHardWareVSkipLines(cbcr_srcH, cbcr_dstH);
                cbcr_yscl_factor = GET_SCALE_FACTOR_BILI_DN_VSKIP(cbcr_srcH, cbcr_dstH,
cbcr_vScaleDnMult);
                //printf("[hxx_dbg] cbcr_vScaleDnMult=%d;\n",cbcr_vScaleDnMult);
                if(cbcr_vScaleDnMult == 4) {
                    cbcr_vsd_bil_gt4 = 1;
                    cbcr_vsd_bil_gt2 = 0;
                } else if(cbcr_vScaleDnMult == 2) {
                    cbcr_vsd_bil_gt4 = 0;
                    cbcr_vsd_bil_gt2 = 1;
                } else {

```

```

        cbcr_vsd_bil_gt4 = 0;
        cbcr_vsd_bil_gt2 = 0;
    }
    break;
case SCALE_DOWN_AVG:
    cbcr_yscl_factor = GET_SCALE_FACTOR_AVRG(cbcr_srcH, cbcr_dstH);
    break;
default :
    break;
} //p_win_para->cbr_vsd_mode
break;

default :
    break;
} //p_win_para->cbr_hor_scl_mode

p_win_para->vsd_cbr_gt4      = cbcr_vsd_bil_gt4;
p_win_para->vsd_cbr_gt2      = cbcr_vsd_bil_gt2;
p_win_para->win_h_Cbr_factor = cbcr_xscl_factor;
p_win_para->win_v_Cbr_factor = cbcr_yscl_factor;

//-----
switch(p_win_para->yrgb_hor_scl_mode) {
    case SCALE_NONE :
        printf("[hxx_dbg] X YRGB SCALE_NONE\n");
        break;
    case SCALE_UP :
        printf("[hxx_dbg] X YRGB SCALE_UP_BICUBIC; yrgb_xscl_factor=%04x;\n",yrgb_xscl_factor);
        //switch(p_win_para->yrgb_hsu_mode) {
        //    case SCALE_UP_BIL :
        //        printf("[hxx_dbg] X YRGB SCALE_UP_BILINEAR;[ERROR] yrgb_xscl_factor=%04x;\n
",yrgb_xscl_factor);
        //        break;
        //    case SCALE_UP_BIC :
        //        printf("[hxx_dbg] X YRGB SCALE_UP_BICUBIC; yrgb_xscl_factor=%04x;\n
",yrgb_xscl_factor);
        //        break;
        //    default :
        //}
        break;
    case SCALE_DOWN :
        switch(p_win_para->yrgb_hsd_mode) {
            case SCALE_DOWN_BIL :
                printf("[hxx_dbg] X YRGB SCALE_DOWN_BILINEAR;
yrgb_xscl_factor=%04x;\n",yrgb_xscl_factor);
                break;
            case SCALE_DOWN_AVG :
                printf("[hxx_dbg] X YRGB SCALE_DOWN_AVERAGE;
yrgb_xscl_factor=%04x;\n",yrgb_xscl_factor);
                break;
            default:
                break;
        }
        break;
    default:
        break;
}

switch(p_win_para->yrgb_ver_scl_mode) {
    case SCALE_NONE :
        printf("[hxx_dbg] Y YRGB SCALE_NONE\n");
        break;
    case SCALE_UP :
        switch(p_win_para->yrgb_vsu_mode) {
            case SCALE_UP_BIL :
                printf("[hxx_dbg] Y YRGB SCALE_UP_BILINEAR;
yrgb_yscl_factor=%04x;\n",yrgb_yscl_factor);
                break;
            default:
                break;
        }
        break;
}

```

```

        break;
    case SCALE_UP_BIC :
        printf("[hxx_dbg] Y YRGB SCALE_UP_BICUBIC;
yrgb_yscl_factor=%04x;\n",yrgb_yscl_factor);
        break;
    default :
        break;
    }
    break;
case SCALE_DOWN :
    switch(p_win_para->yrgb_vsd_mode) {
    case SCALE_DOWN_BIL :
        printf("[hxx_dbg] Y YRGB SCALE_DOWN_BILINEAR;
yrgb_yscl_factor=%04x;\n",yrgb_yscl_factor);
        break;
    case SCALE_DOWN_AVG :
        printf("[hxx_dbg] Y YRGB SCALE_DOWN_AVERAGE;
yrgb_yscl_factor=%04x;\n",yrgb_yscl_factor);
        break;
    default:
        break;
    }
    break;
default:
    break;
}
//-----
switch(p_win_para->cbr_hor_scl_mode) {
    case SCALE_NONE :
        printf("[hxx_dbg] X CBCR SCALE_NONE\n");
        break;
    case SCALE_UP :
        printf("[hxx_dbg] X CBCR SCALE_UP_BICUBIC; cbcr_xscl_factor=%04x;\n",cbcr_xscl_factor);
        //switch(p_win_para->cbr_hsu_mode) {
        //    printf("[hxx_dbg] X CBCR SCALE_UP_BICUBIC; cbcr_xscl_factor=%04x;\n
",cbcr_xscl_factor);
        //    case SCALE_UP_BIL :
        //        printf("[hxx_dbg] X CBCR SCALE_UP_BILINEAR;[ERROR] cbcr_xscl_factor=%04x;\n
",cbcr_xscl_factor);
        //        break;
        //    case SCALE_UP_BIC :
        //        printf("[hxx_dbg] X CBCR SCALE_UP_BICUBIC; cbcr_xscl_factor=%04x;\n
",cbcr_xscl_factor);
        //        break;
        //    default :
        //}
        break;
    case SCALE_DOWN :
        switch(p_win_para->cbr_hsd_mode) {
        case SCALE_DOWN_BIL :
            printf("[hxx_dbg] X CBCR SCALE_DOWN_BILINEAR;
cbcr_xscl_factor=%04x;\n",cbcr_xscl_factor);
            break;
        case SCALE_DOWN_AVG :
            printf("[hxx_dbg] X CBCR SCALE_DOWN_AVERAGE;
cbcr_xscl_factor=%04x;\n",cbcr_xscl_factor);
            break;
        default:
            break;
        }
        break;
    }
    break;
default:
    break;
}

switch(p_win_para->cbr_ver_scl_mode) {
    case SCALE_NONE :

```

```

printf("[hxx_dbg] Y CBCR SCALE_NONE\n");
break;
case SCALE_UP :
switch(p_win_para->cbr_vsu_mode) {
    case SCALE_UP_BIL :
        printf("[hxx_dbg] Y CBCR SCALE_UP_BILINEAR;
cbcr_yscl_factor=%04x;\n",cbcr_yscl_factor);
        break;
    case SCALE_UP_BIC :
        printf("[hxx_dbg] Y CBCR SCALE_UP_BICUBIC;
cbcr_yscl_factor=%04x;\n",cbcr_yscl_factor);
        break;
    default :
        break;
}
break;
case SCALE_DOWN :
switch(p_win_para->cbr_vsd_mode) {
    case SCALE_DOWN_BIL :
        printf("[hxx_dbg] Y CBCR SCALE_DOWN_BILINEAR;
cbcr_yscl_factor=%04x;\n",cbcr_yscl_factor);
        break;
    case SCALE_DOWN_AVG :
        printf("[hxx_dbg] Y CBCR SCALE_DOWN_AVERAGE;
cbcr_yscl_factor=%04x;\n",cbcr_yscl_factor);
        break;
    default:
        break;
}
break;
default:
break;
}

//-----
//printf("[hxx_dbg] p_win_para->yrgb_hor_scl_mode=%d;\n",p_win_para->yrgb_hor_scl_mode);
//printf("[hxx_dbg] p_win_para->yrgb_ver_scl_mode=%d;\n",p_win_para->yrgb_ver_scl_mode);
//printf("[hxx_dbg] p_win_para->cbcr_hor_scl_mode=%d;\n",p_win_para->cbcr_hor_scl_mode );
//printf("[hxx_dbg] p_win_para->cbcr_ver_scl_mode=%d;\n",p_win_para->cbcr_ver_scl_mode );

//printf("[hxx_dbg] p_win_para->yrgb_hsd_mode= %d;\n",p_win_para->yrgb_hsd_mode);
//printf("[hxx_dbg] p_win_para->cbr_hsd_mode = %d;\n",p_win_para->cbr_hsd_mode );

//printf("[hxx_dbg] p_win_para->yrgb_vsu_mode= %d;\n",p_win_para->yrgb_vsu_mode);
//printf("[hxx_dbg] p_win_para->yrgb_vsd_mode= %d;\n",p_win_para->yrgb_vsd_mode);
//printf("[hxx_dbg] p_win_para->cbr_vsu_mode = %d;\n",p_win_para->cbr_vsu_mode );
//printf("[hxx_dbg] p_win_para->cbr_vsd_mode = %d;\n",p_win_para->cbr_vsd_mode );

//printf("[hxx_dbg] yrgb_xscl_factor= %04x\n", yrgb_xscl_factor);
//printf("[hxx_dbg] yrgb_yscl_factor= %04x\n", yrgb_yscl_factor);
//
//printf("[hxx_dbg] cbcr_xscl_factor= %04x\n", cbcr_xscl_factor);
//printf("[hxx_dbg] cbcr_yscl_factor= %04x\n", cbcr_yscl_factor);

} //end calc_win_scl_factor

```

## 2.Limitation of YUV scaling down(3840 < width < 4096)

Limitation of 3840~4096 horizontal scale down for YUV422/420:

### YUV422

- (1) not support vertical scale up/down if width(>3840) scale down to width(>2560);
- (2) support vertical down but only support vertical bilinear scale up if width(>3840) scale down to width(>1920 and <2560);

(3) no limitation if width(>3840) scale down to width(<=1920);

### **YUV420**

- (1) not support width(>3840) scale down to width(>2560);
- (2) support vertical down but only support vertical bilinear scale up if width(>3840) scale down to width(>1920 and <2560);
- (3) no limitation if width(>3840) scale down to width(<=1920);

Since the sampling rate is different for lumina data and chroma data with the format of YCbCr422 and YCbCr420, the scaling factors for lumina data and chroma data are calculated and configured in VOP\_WIN0\_SCL\_FACTOR\_Y/ VOP\_WIN0\_SCL\_FACTOR\_CBR respectively.

### **27.3.4 De-flicker**

It is necessary to display a non-interlaced video signal on an interlaced display panel (such as TV set). Thus "non-interlaced-to-interlaced conversion" is required.

The easiest approach is to throw away every other active scan line in each non-interlaced frame. Although the cost is minimal, there are problems with this approach. If there is a sharp vertical transition of color or intensity, it will flicker at one-half the refresh rate.

A better solution is to use two lines of non-interlaced data to generate one line of interlace data. Fast vertical transition is smoothed out over several interlace lines.

The vertical filtering of two non-interlaced lines can be done by enabling the vertical scaling offset updated dynamically in different fields, i.e, even field and odd field. The dynamic updated value of scaling offset is half of the scaling factor.

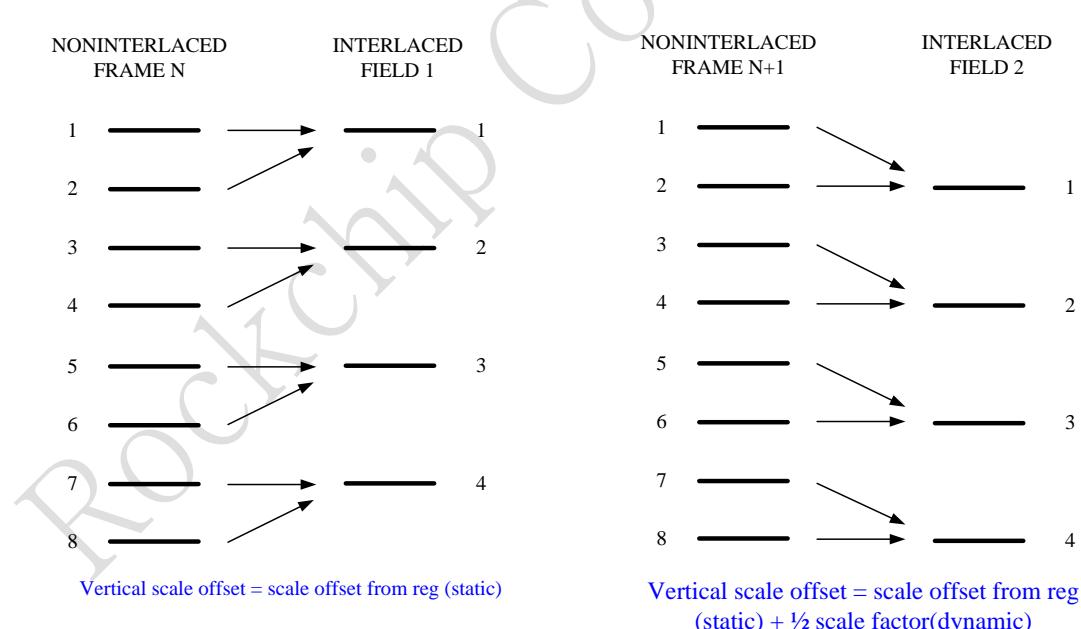


Fig. 27-7 De-flicker

### **27.3.5 Virtual display**

When in virtual display, the active image is part of the virtual (original) image in frame buffer memory.

The virtual width is indicated by setting VIR\_STRIDE and VIR\_STRIDE for different data format. Note that RGB/BPP has one stride——yrgb\_vir\_stride; YUV has two virtual stride——yrgb\_vir\_stride and cbcr\_vir\_stride.

For RGB-8bit and YUV-8bit, the stride should be multiples of word (32-bit), with dummy bytes in the end of virtual line if the original width is not 32-bit aligned.

For YUV-10bit, the stride should be multiples of word (-bit), with dummy bytes in the end of virtual line if the original width is not 128-bit aligned.

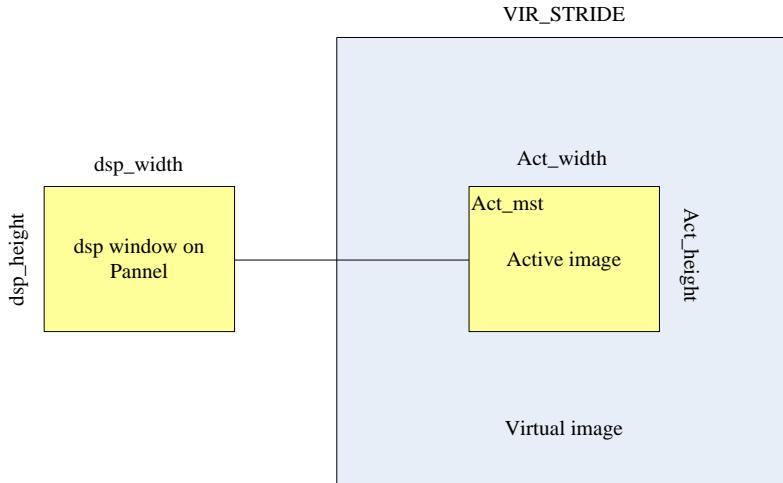


Fig. 27-8 Virtual display

### 27.3.6 MIRROR display

Mirror display is necessary for the panel with mirror timing interface. There are two types of mirror mode: horizontal mirror(X-Mirror) and vertical mirror(Y-mirror).

The default display order is from left to right(L2R) in horizontal direction and from top to bottom(T2B) in vertical direction. However, when X-Mirror is enable, the horizontal display order is from right to left(R2L); when Y-MIRROR is enable, the vertical display order is from bottom to top(B2T).

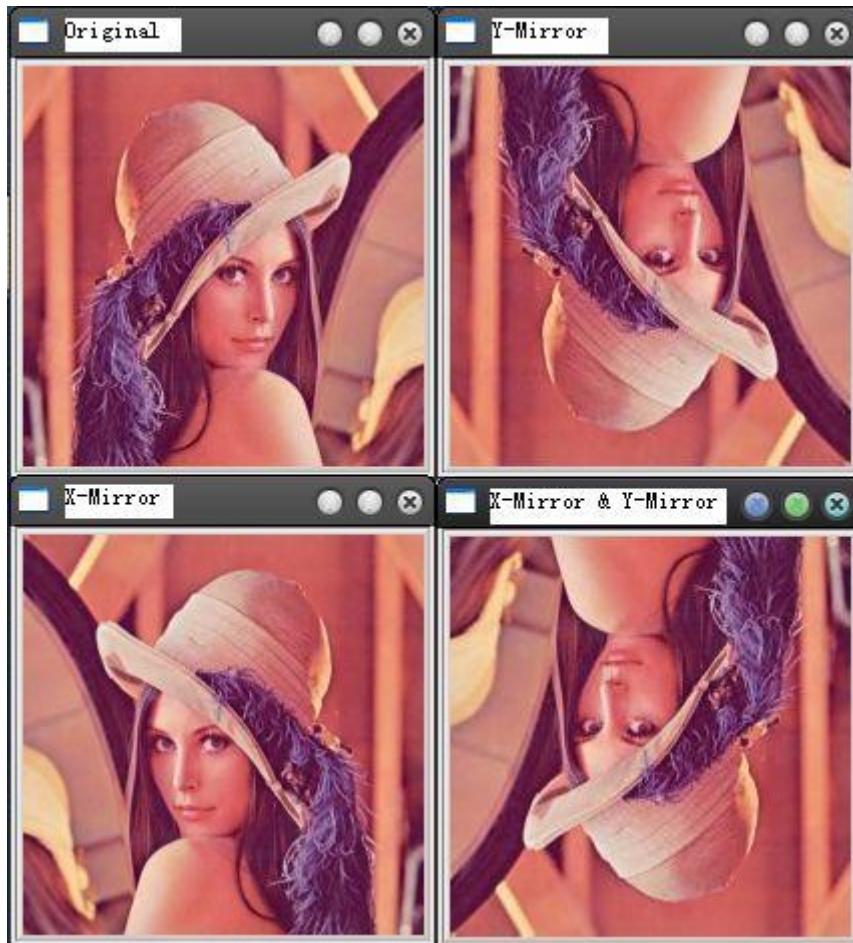
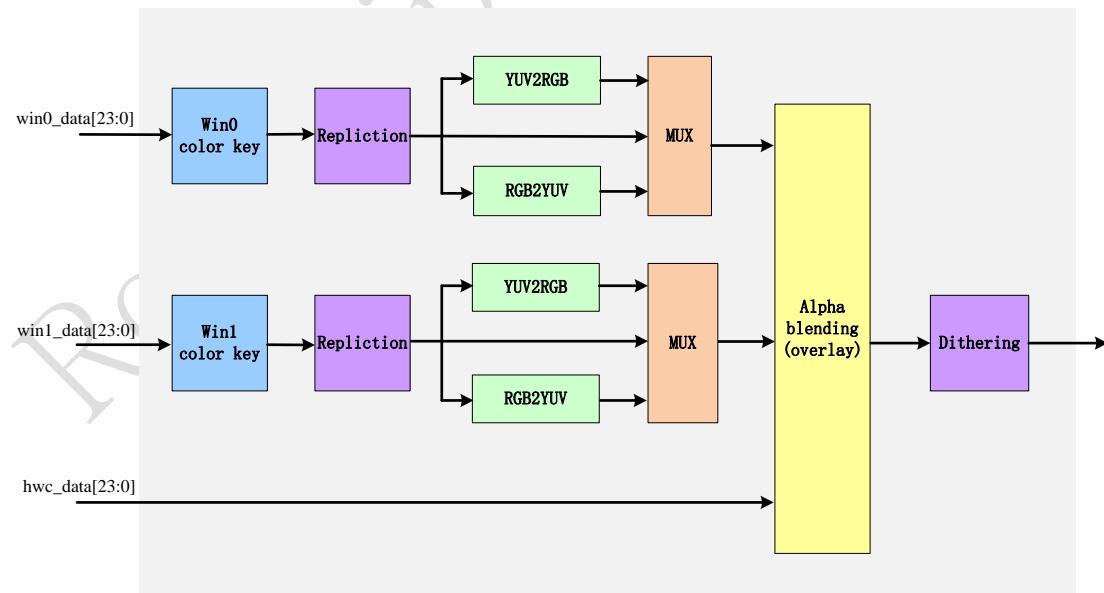


Fig. 27-9 X-Mirror and Y-Mirror

### 27.3.7 Display process



#### 1. Overlay display

There are totally 4 layers for overlay display: Background, win0 layer, win1 layer and hardware cursor layer(HWC).

Background is a programmable solid color layer, which is always in the bottom of the display screen.

HWC is a 32x32 or 64x64 3-LUT-colors layer, which is always on the top of the display screen.

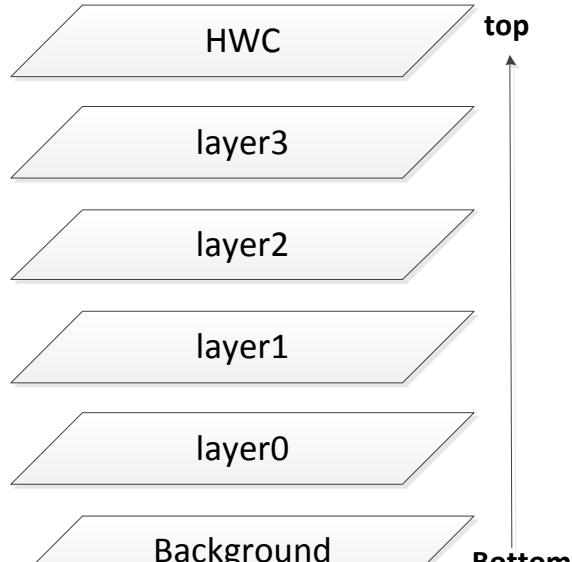
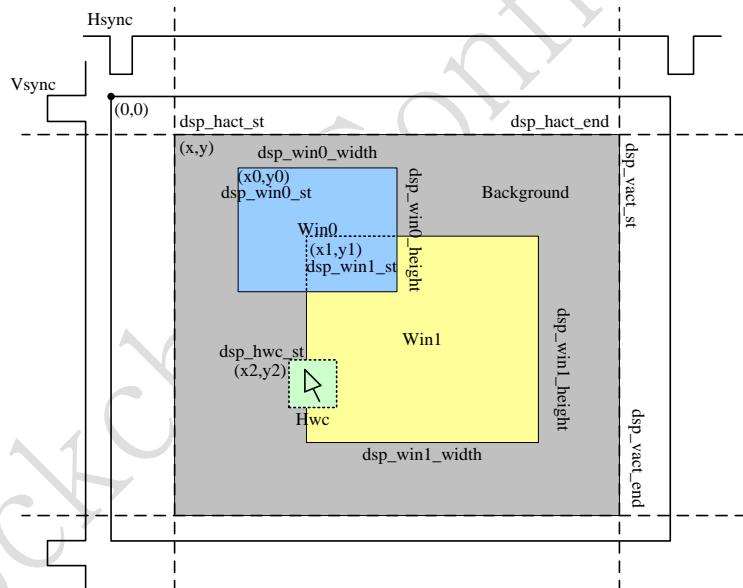


Fig. 27-10 overlay

Following figure is an example of overlay display for win0,win1 and hwc.



## 2.Post scale down

Post scale down after overlay is supported to fix overscan, that draws the borders of the image beyond the normally visible area on the screen.

The scale ratio of post scale down is 0.5~1.

### Post timing setting

The post scale parameter ,such as,post\_dsp\_hact\_st,post\_dsp\_hact\_end, post\_dsp\_vact\_st,post\_dsp\_vact\_end can be configured.

When post scaling equal "1" ,the post scaler parameter are the same as dsp timing parameter.

eg:

```

post_dsp_hact_st = dsp_hact_st
post_dsp_hact_end = dap_hact_end
post_dsp_vact_st = dsp_vact_st
post_dsp_vact_end = dsp_vact_end

```

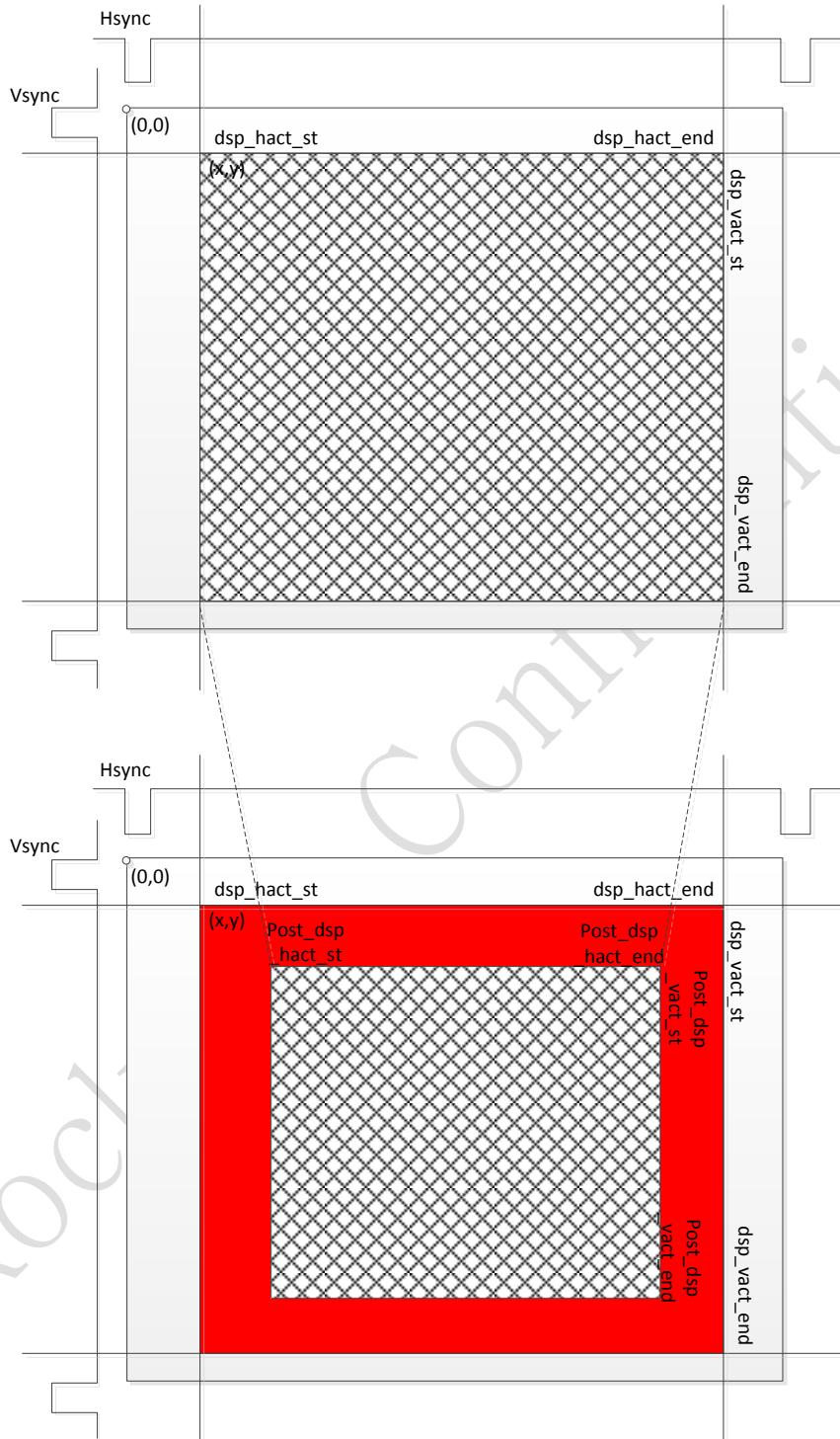


Fig. 27-11 post scaling timing

### Post scale down factor

For horizontal scale down,factor =  $((src\_width*2/3)<<16)/(dst\_width-1)$ .

For vertical scale down,factor =  $((src\_width*2/3)<<16)/(dst\_width-1)$ .

### 3.Transparency color key

The transparency color key value defines the pixel treated as transparent pixel. The pixel whose value is equal to the color key value could not be visible on the screen, instead of the pixel in the under layer or solid background color.

There are two transparency color key for win0 layer and win1 layer respectively. When color key is enable, the transparency process is done after scaling but before YUV2RGB color space converter.

Moreover, transparency color key is just available for non-scaling mode.

Following figure is an example of transparency color key for win0 and win1.

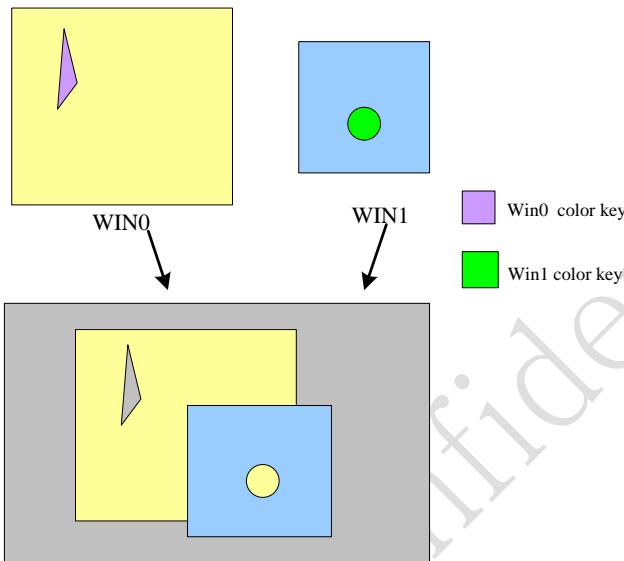
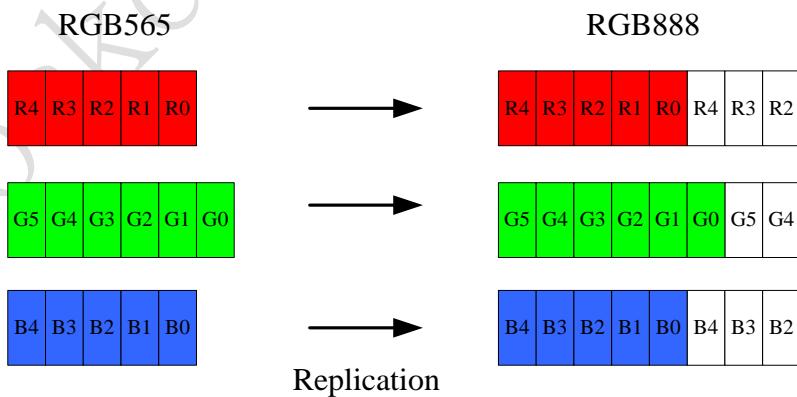


Fig. 27-12 Transparency Color Key

#### 4.Replication(dither up)

If the size of panel data bus is larger than the size of source pixel data, i.e., the source input format is RGB565 and display output format is RGB888, you could do bit replication by replicating MSBs to LSBs if replication is enable (VOP\_DSP\_CTRL0[9]=1) or filling with "0" to LSBs if replication is disable (VOP\_DSP\_CTRL0[9]=0).



#### 5.Alpha blending

There are 12 alpha blending mode between two overlay layers for layer1/layer2/layer3/hwc. Layer0 does not support alpha blending with background.

When in per-pixel mode, the alpha value for every pixel is following with the pixel data. i.e., aRGB, and can be scaled like RGB data. Therefore it is just suitable for win0/win1/win2/win3/hwc layer with ARGB data format.

The alpha blending architecture is shown as follows.

Table 27-1 alpha blending mode settings

<b>Blending Mode</b>	<b>Cs'</b>	<b>Fs</b>	<b>Cd'</b>	<b>Fd</b>
AA_USER_DEFINED	X	User defined	Cd	User defined
AA_CLEAR	X	0	Cd	0
AA_SRC	X	0	Cd	1
AA_DST	X	1	Cd	1
AA_SRC_OVER	Cs	1	Cd	1-As''
AA_DST_OVER	Cs	1-As''	Cd	1
AA_SRC_IN	Cs	As''	Cd	0
AA_DST_IN	X	0	Cd	As''
AA_SRC_OUT	Cs	1-As''	Cd	0
AA_DST_OUT	X	0	Cd	1-As''
AA_SRC_ATOP	Cs	As''	Cd	1-As''
AA_DST_ATOP	Cs	1-As''	Cd	As''
AA_XOR	Cs	1-As''	Cd	1-As''
AA_SRC_OVER_GLOBAL	Cs*As''	Ags''	Cd	1-As''

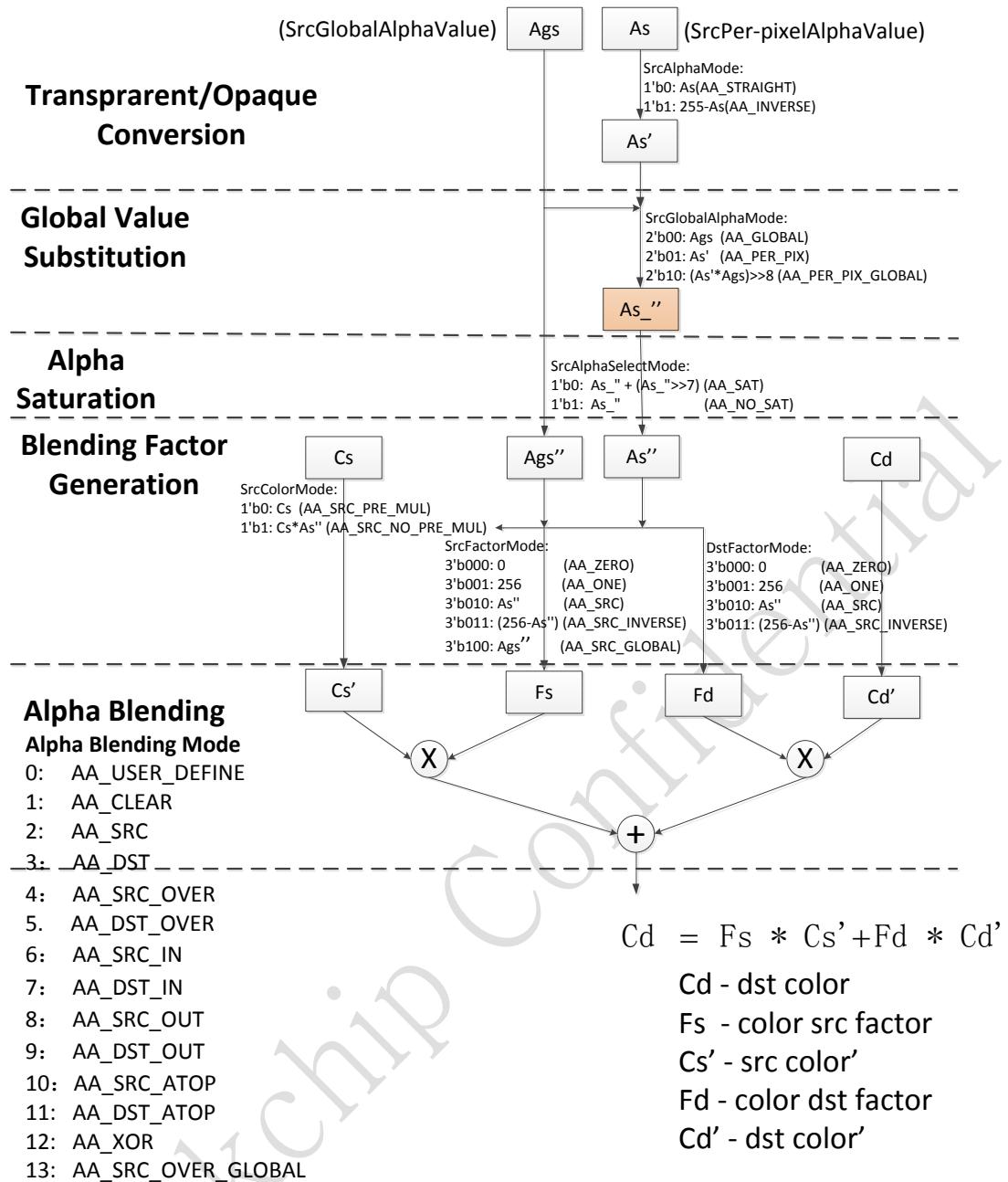


Fig. 27-13 alpha configuration flow

Pseudo Code:

```

switch(alpha_config->alpha_blending_mode)
{
    case AA_USER_DEFINE:
        break;
    case AA_CLEAR:
        alpha_config->src_factor_mode=AA_ZERO;
        alpha_config->dst_factor_mode=AA_ZERO;
        break;
    case AA_SRC:
        alpha_config->src_factor_mode=AA_ONE;
        alpha_config->dst_factor_mode=AA_ZERO;
        break;
    case AA_DST:
        alpha_config->src_factor_mode=AA_ZERO;
        alpha_config->dst_factor_mode=AA_ONE;
        break;
}

```

```

case AA_SRC_OVER:
alpha_config->src_color_mode=AA_SRC_PRE_MUL;
alpha_config->src_factor_mode=AA_ONE;
alpha_config->dst_factor_mode=AA_SRC_INVERSE;
break;
case AA_DST_OVER:
alpha_config->src_color_mode=AA_SRC_PRE_MUL;
alpha_config->src_factor_mode=AA_SRC_INVERSE;
alpha_config->dst_factor_mode=AA_ONE;
break;
case AA_SRC_IN:
alpha_config->src_color_mode=AA_SRC_PRE_MUL;
alpha_config->src_factor_mode=AA_SRC;
alpha_config->dst_factor_mode=AA_ZERO;
break;
case AA_DST_IN:
alpha_config->src_factor_mode=AA_ZERO;
alpha_config->dst_factor_mode=AA_SRC;
break;
case AA_SRC_OUT:
alpha_config->src_color_mode=AA_SRC_PRE_MUL;
alpha_config->src_factor_mode=AA_SRC_INVERSE;
alpha_config->dst_factor_mode=AA_ZERO;
break;
case AA_DST_OUT:
alpha_config->src_factor_mode=AA_ZERO;
alpha_config->dst_factor_mode=AA_SRC_INVERSE;
break;
case AA_SRC_ATOP:
alpha_config->src_color_mode=AA_SRC_PRE_MUL;
alpha_config->src_factor_mode=AA_SRC;
alpha_config->dst_factor_mode=AA_SRC_INVERSE;
break;
case AA_DST_ATOP:
alpha_config->src_color_mode=AA_SRC_PRE_MUL;
alpha_config->src_factor_mode=AA_SRC_INVERSE;
alpha_config->dst_factor_mode=AA_SRC;
break;
case AA_XOR:
alpha_config->src_color_mode=AA_SRC_PRE_MUL;
alpha_config->src_factor_mode=AA_SRC_INVERSE;
alpha_config->dst_factor_mode=AA_SRC_INVERSE;
break;
case AA_SRC_OVER_GLOBAL:
alpha_config->src_global_alpha_mode=AA_PER_PIX_GLOBAL;
alpha_config->src_color_mode=AA_SRC_NO_PRE_MUL;
alpha_config->src_factor_mode=AA_SRC_GLOBAL;
alpha_config->dst_factor_mode=AA_SRC_INVERSE;
break;
default:
printf("alpha mode error\n");
break;
}

```

**6.CABC**

CABC(Content Adaptive Backlight Control) is used to increase the contrast of such LCD-screens the backlight can be (globally) dimmed when the image to be displayed is dark (i.e. not comprising high intensity image data) while the image data is numerically corrected and adapted to the reduced backlight intensity.

Config the panel total pixel num to reg 0x1c4 ,and config the calc pixel num to regfile 0x1c0 (typical calc\_pixel\_num / total\_pixel\_num  $\approx$  80% ~90%).

Config the stage up and stage down to ensure the luminance difference will not be too big in each two adjacent frames. Typical value is 0x20~0x40.

There are 3x7 Gaussian filter tables in reg 0x1c8~0x1dc.

default value as follow:

0x1c8 : 0x15110903

0x1cc : 0x00030911

0x1d0 : 0x1a150b04

0x1d4 : 0x00040b15

0x1d8 : 0x15110903

0x1dc : 0x00030911

## 7.BCSH

BCSH is used to adjust "Brightness,Contrast,Saturation,Hue,like IEP BCSH-8bit. For details,please refer to IEP chapter.

- Extend yuv data from 8bits(IEP) to 10bits.
- The brightness adjust support (-128,127).
- The yuv data of color bar are 10bits.

## 8.Color space conversion

There are three standards for YUV2RGB-8bit, and BT2020 standard for YUV2RGB-10bit.

- YUV2RGB-8bit

1. yuv to rgb (REC-601) range 0 (Y[16:235], UV[16:240], RGB[0:255])

$$R = 1.164(Y-16) + 1.596(V-128)$$

$$G = 1.164(Y-16) - 0.391(U-128) - 0.813(V-128)$$

$$B = 1.164(Y-16) + 2.018(U-128)$$

2. yuv to rgb (REC-601) range 1 (YUV[0:255], RGB[0:255])

$$R = (Y-16) + 1.402(V-128)$$

$$G = (Y-16) - 0.344(U-128) - 0.714(V-128)$$

$$B = (Y-16) + 1.772(U-128)$$

3. yuv to rgb (REC-709) range 0 (Y[16:235], UV[16:240], RGB[0:255])

$$R = 1.164(Y-16) + 1.793(V-128)$$

$$G = 1.164(Y-16) - 0.213(U-128) - 0.534(V-128)$$

$$B = 1.164(Y-16) + 2.115(U-128)$$

- RGB2YUV-8bit

ccir601

$$Y = 0.257R + 0.504G + 0.098B + 16$$

$$Cb = -0.148R - 0.291G + 0.439B + 128$$

$$Cr = 0.439R - 0.368G - 0.071B + 128$$

- YUV2RGB-10bit

$$Y = (230R+595G+52B+65536)/1024$$

$$U = (-125R - 323G + 449B + 524288)/1024$$

$$V = (449R - 412G - 36B + 524288)/1024$$

- RGB2YUV-10bit

$$R = 1.1636Y + 1.6778V - 933.504$$

$$G = 1.1636Y - 0.1872U - 0.6501V + 351.9232$$

$$B = 1.1636Y + 2.1406U - 1170.4576$$

## 9.Dither Down

### Dither down directly

The invalid lower bits will be replaced by "0" after dither operation, if disable dither down. eg: 10'b10\_1011\_00XX → 10'b10\_1011\_0000.

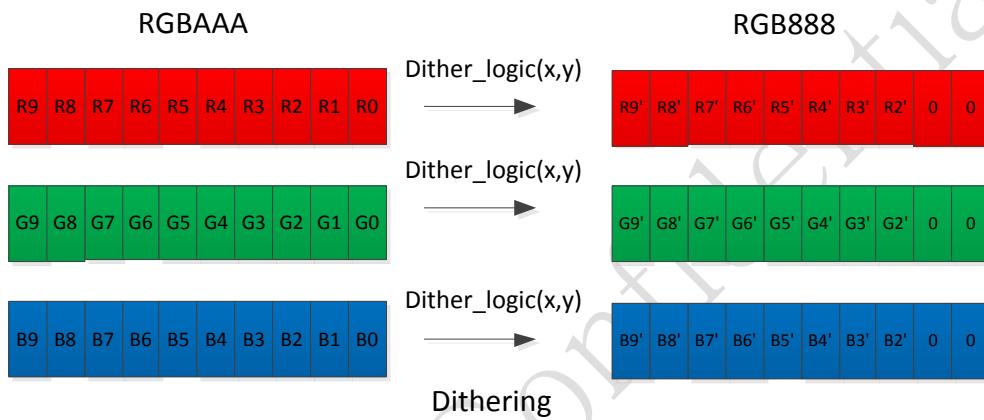


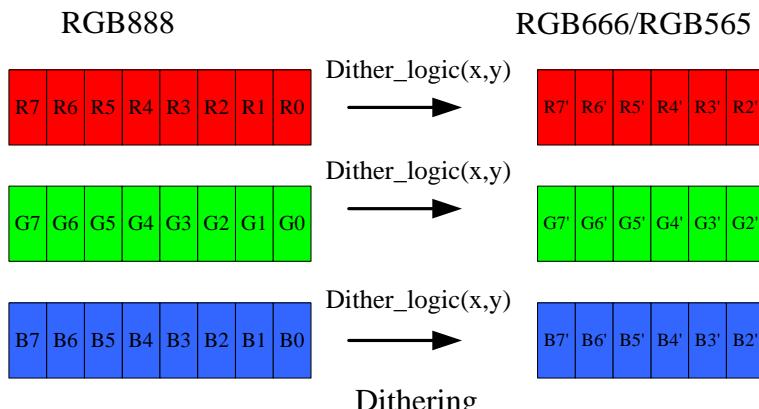
Fig. 27-14 Dither down directly

### Allegro Dither Down

Dithering is an intentional applied form of noise, using to randomize quantization error, and thereby preventing large-scaling patterns such as "banding".

The pixel value is used by dithering process to display the data in a lower color depth on the LCD panel, i.e, the source input format is RGB888 and display output format is RGB565 or RGB666. When dithering is enable(VOP\_DSP\_CTRL0[11]=1), the output data is generated by dithering algorithm based on the pixel position and the value of removed bits. Otherwise, the MSBs of the pixel color components are output as display data.

There are two dither modes: "RGB888 to RGB666" and "RGB888 to RGB565", which is defined by VOP\_DSP\_CTRL0[10]. When VOP\_DSP\_CTRL0[10] is 1, dithering with "RGB888 to RGB666" is available; otherwise, "RGB888 to RGB565" is used.



## FRC Dither Down

The pattern of dither frc was configured by vop regfile vop\_base+0x1e0~0x1f4.

The following fig is the default pattern picture in vop, you can config different value of regfile 0x1e0~0x1f4, to change the pattern picture.

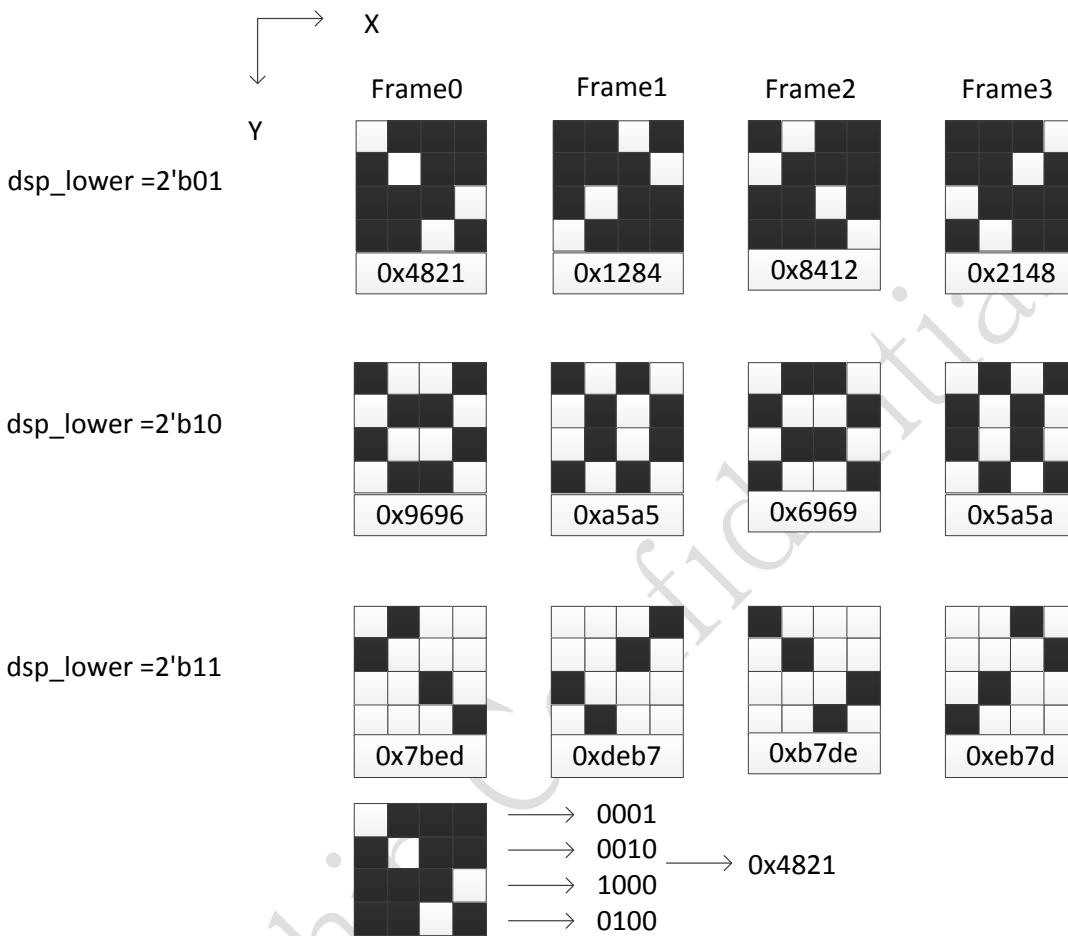


Fig. 27-15 frc pattern diagram

There are four typical pattern as follow :

(1) default pattern:

`0x1e0 : 0x12844821`  
`0x1e4 : 0x21488412`  
`0x1e8 : 0xa55a9696`  
`0x1ec : 0x5aa56969`  
`0x1f0 : 0xdeb77bed`  
`0x1f4 : 0xed7bb7de`

(2) for column inversion panel:

`0x1e0 : 0x50a00a05`  
`0x1e4 : 0xa050050a`  
`0x1e8 : 0x5a5aa5a5`  
`0x1ec : 0x5a5aa5a5`  
`0x1f0 : 0xaf5ff5fa`  
`0x1f4 : 0x5ffffaf5`

(3) for 1+2dot panel

`0x1e0 : 0x0c308421`

0x1e4 : 0x124803c0  
 0x1e8 : 0xcc339669  
 0x1ec : 0x33cc9669  
 0x1f0 : 0xf3cf7bde  
 0x1f4 : 0xedb7fc3f

(4) default enhance pattern

0x1e0 : 0x12844821  
 0x1e4 : 0x21488412  
 0x1e8 : 0x55aaaa55  
 0x1ec : 0x55aaaa55  
 0x1f0 : 0xdeb77bed  
 0x1f4 : 0xed7bb7de

## 10.Gamma Correction

Gamma Correction is necessary because most monitors don't have a linear relationship between the voltage and the brightness, which results in your scene looking like it has too much contrast and the light falling off from the source outward, happens too quickly. The result can also be problematic if you are going into a composition program.

You can correct this by "Gamma Correction", which allows you to display the images and textures on your computer in an accurate manner.

Your screen is not linear, in that it displays the brightness unevenly. As a result, the image looks to be more high contrast than it should, you end up adding more lights or turning up the intensity, or you don't use the lighting in a realistic way that matches well with live action scenes. It also creates problems for you if you use compositing software.

There are three 1024x10bits line buffers separately for 10bit-R/G/B gamma correction. For 8bit-RGB, it only consumes 256x8bit for each channel. You can write gamma correction LUT through register "GAMMA\_LUT\_ADDR" one by one.

## 11.Output format

Config `dsp_out_mode` register to adapt a variety of panel interface. As follow:



Fig. 27-16 `dsp_out_mode` description

## 27.4 Register Description

### 27.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
VOP_REG_CFG_DONE	0x0000	W	0x00000000	Register config done flag
VOP_VERSION_INFO	0x0004	W	0x00000000	
VOP_SYS_CTRL	0x0008	W	0x00801000	System control register0

Name	Offset	Size	Reset Value	Description
VOP_SYS_CTRL1	0x000c	W	0x00000000	
VOP_DSP_CTRL0	0x0010	W	0x00000000	Display control register0
VOP_DSP_CTRL1	0x0014	W	0x0000e400	Display control register1
VOP_DSP_BG	0x0018	W	0x00000000	background color
VOP MCU_CTRL	0x001c	W	0x00711c08	MCU mode control register
VOP_INTR_CTRL0	0x0020	W	0x00000000	Interrupt ctrl register0
VOP_INTR_CTRL1	0x0024	W	0x00000000	Interrupt ctrl register1
VOP_INTR_RESERVED0	0x0028	W	0x00000000	
VOP_INTR_RESERVED1	0x002c	W	0x00000000	
VOP_WIN0_CTRL0	0x0030	W	0x00000040	win0 ctrl register0
VOP_WIN0_CTRL1	0x0034	W	0x00000000	win1 ctrl register1
VOP_WIN0_COLOR_KEY	0x0038	W	0x00000000	Win0 color key register
VOP_WIN0_VIR	0x003c	W	0x00000140	Win0 virtual stride
VOP_WIN0_YRGB_MST	0x0040	W	0x00000000	Win0 YRGB memory start address
VOP_WIN0_CBR_MST	0x0044	W	0x00000000	Win0 Cbr memory start address
VOP_WIN0_ACT_INFO	0x0048	W	0x00ef013f	Win0 active window width/height
VOP_WIN0_DSP_INFO	0x004c	W	0x00ef013f	Win0 display width/height on panel
VOP_WIN0_DSP_ST	0x0050	W	0x000a000a	Win0 display start point on panel
VOP_WIN0_SCL_FACTO R_YRGB	0x0054	W	0x10001000	Win0 YRGB scaling factor
VOP_WIN0_SCL_FACTO R_CBR	0x0058	W	0x10001000	Win0 Cbr scaling factor
VOP_WIN0_SCL_OFFSET	0x005c	W	0x00000000	Win0 scaling start point offset
VOP_WIN0_SRC_ALPHA _CTRL	0x0060	W	0x00000000	
VOP_WIN0_DST_ALPHA _CTRL	0x0064	W	0x00000000	
VOP_WIN0_FADEING_CT RL	0x0068	W	0x00000000	
VOP_WIN0_RESERVED0	0x006c	W	0x00000000	
VOP_WIN1_CTRL0	0x0070	W	0x00000040	win1 ctrl register0
VOP_WIN1_CTRL1	0x0074	W	0x00000000	win1 ctrl register1

Name	Offset	Size	Reset Value	Description
VOP_WIN1_COLOR_KEY	0x0078	W	0x00000000	Win1 color key register
VOP_WIN1_VIR	0x007c	W	0x00000140	win1 virtual stride
VOP_WIN1_YRGB_MST	0x0080	W	0x00000000	Win1 YRGB memory start address
VOP_WIN1_CBR_MST	0x0084	W	0x00000000	Win1 Cbr memory start address
VOP_WIN1_ACT_INFO	0x0088	W	0x00ef013f	Win1 active window width/height
VOP_WIN1_DSP_INFO	0x008c	W	0x00ef013f	Win1 display width/height on panel
VOP_WIN1_DSP_ST	0x0090	W	0x000a000a	Win1 display start point on panel
VOP_WIN1_SCL_FACTO R_YRGB	0x0094	W	0x10001000	Win1 YRGB scaling factor
VOP_WIN1_SCL_FACTO R_CBR	0x0098	W	0x10001000	Win1 Cbr scaling factor
VOP_WIN1_SCL_OFFSET	0x009c	W	0x00000000	Win1 scaling start point offset
VOP_WIN1_SRC_ALPHA _CTRL	0x00a0	W	0x00000000	
VOP_WIN1_DST_ALPHA _CTRL	0x00a4	W	0x00000000	
VOP_WIN1_FADEDING_CT RL	0x00a8	W	0x00000000	
VOP_WIN1_RESERVED0	0x00ac	W	0x00000000	
VOP_WIN2_CTRL0	0x00b0	W	0x00000000	win2 ctrl register0
VOP_WIN2_CTRL1	0x00b4	W	0x00000000	win2 ctrl register0
VOP_WIN2_VIR0_1	0x00b8	W	0x01400140	Win2 virtual stride0 and virtaul stride1
VOP_WIN2_VIR2_3	0x00bc	W	0x01400140	Win2 virtual stride2 and virtaul stride3
VOP_WIN2_MST0	0x00c0	W	0x00000000	Win2 memory start address0
VOP_WIN2_DSP_INFO0	0x00c4	W	0x00ef013f	Win2 display width0/height0 on panel
VOP_WIN2_DSP_ST0	0x00c8	W	0x000a000a	Win2 display start point0 on panel
VOP_WIN2_COLOR_KEY	0x00cc	W	0x00000000	Win2 color key register
VOP_WIN2_MST1	0x00d0	W	0x00000000	Win2 memory start address1

Name	Offset	Size	Reset Value	Description
VOP_WIN2_DSP_INFO1	0x00d4	W	0x00ef013f	Win2 display width1/height1 on panel
VOP_WIN2_DSP_ST1	0x00d8	W	0x000a000a	Win2 display start point1 on panel
VOP_WIN2_SRC_ALPHA_CTRL	0x00dc	W	0x00000000	
VOP_WIN2_MST2	0x00e0	W	0x00000000	Win2 memory start address2
VOP_WIN2_DSP_INFO2	0x00e4	W	0x00ef013f	Win2 display width2/height2 on panel
VOP_WIN2_DSP_ST2	0x00e8	W	0x000a000a	Win2 display start point2 on panel
VOP_WIN2_DST_ALPHA_CTRL	0x00ec	W	0x00000000	
VOP_WIN2_MST3	0x00f0	W	0x00000000	Win2 memory start address3
VOP_WIN2_DSP_INFO3	0x00f4	W	0x00ef013f	Win2 display width3/height3 on panel
VOP_WIN2_DSP_ST3	0x00f8	W	0x000a000a	Win2 display start point3 on panel
VOP_WIN2_FADE_CTRL	0x00fc	W	0x00000000	
VOP_WIN3_CTRL0	0x0100	W	0x00000000	win0 ctrl register0
VOP_WIN3_CTRL1	0x0104	W	0x00000000	win0 ctrl register1
VOP_WIN3_VIRO_1	0x0108	W	0x01400140	Win3 virtual stride0 and virtaul stride1
VOP_WIN3_VIR2_3	0x010c	W	0x01400140	Win3 virtual stride2 and virtaul stride3
VOP_WIN3_MST0	0x0110	W	0x00000000	Win3 memory start address0
VOP_WIN3_DSP_INFO0	0x0114	W	0x00ef013f	Win3 display width0/height0 on panel
VOP_WIN3_DSP_ST0	0x0118	W	0x000a000a	Win3 display start point0 on panel
VOP_WIN3_COLOR_KEY	0x011c	W	0x00000000	Win3 color key register
VOP_WIN3_MST1	0x0120	W	0x00000000	Win3 memory start address1

Name	Offset	Size	Reset Value	Description
VOP_WIN3_DSP_INFO1	0x0124	W	0x00ef013f	Win3 display width1/height1 on panel
VOP_WIN3_DSP_ST1	0x0128	W	0x000a000a	Win3 display start point1 on panel
VOP_WIN3_SRC_ALPHA_CTRL	0x012c	W	0x00000000	
VOP_WIN3_MST2	0x0130	W	0x00000000	Win3 memory start address2
VOP_WIN3_DSP_INFO2	0x0134	W	0x00ef013f	Win3 display width2/height2 on panel
VOP_WIN3_DSP_ST2	0x0138	W	0x000a000a	Win3 display start point2 on panel
VOP_WIN3_DST_ALPHA_CTRL	0x013c	W	0x00000000	
VOP_WIN3_MST3	0x0140	W	0x00000000	Win3 memory start address3
VOP_WIN3_DSP_INFO3	0x0144	W	0x00ef013f	Win3 display width3/height3 on panel
VOP_WIN3_DSP_ST3	0x0148	W	0x000a000a	Win3 display start point3 on panel
VOP_WIN3_FADEING_CTRL	0x014c	W	0x00000000	
VOP_HWC_CTRL0	0x0150	W	0x00000000	Hwc ctrl register0
VOP_HWC_CTRL1	0x0154	W	0x00000000	Hwc ctrl register1
VOP_HWC_MST	0x0158	W	0x00000000	Hwc memory start address
VOP_HWC_DSP_ST	0x015c	W	0x000a000a	Hwc display start point on panel
VOP_HWC_SRC_ALPHA_CTRL	0x0160	W	0x00000000	
VOP_HWC_DST_ALPHA_CTRL	0x0164	W	0x00000000	
VOP_HWC_FADEING_CTRL	0x0168	W	0x00000000	
VOP_HWC_RESERVED1	0x016c	W	0x00000000	
VOP_POST_DSP_HACT_INFO	0x0170	W	0x000a014a	post scaler down horizontal start and end

Name	Offset	Size	Reset Value	Description
VOP_POST_DSP_VACT_INFO	0x0174	W	0x000a00fa	Panel active horizontal scanning start point and end point
VOP_POST_SCL_FACTO_R_YRGB	0x0178	W	0x10001000	post yrgb scaling factor
VOP_POST_RESERVED	0x017c	W	0x00001000	
VOP_POST_SCL_CTRL	0x0180	W	0x00000000	post scaling start point offset
VOP_POST_DSP_VACT_INFO_F1	0x0184	W	0x000a00fa	Panel active horizontal scanning start point and end point F1
VOP_DSP_HTOTAL_HS-END	0x0188	W	0x014a000a	Panel scanning horizontal width and hsync pulse end point
VOP_DSP_HACT_ST_END	0x018c	W	0x000a014a	Panel active horizontal scanning start point and end point
VOP_DSP_VTOTAL_VS-END	0x0190	W	0x00fa000a	Panel scanning vertical height and vsync pulse end point
VOP_DSP_VACT_ST_END	0x0194	W	0x000a00fa	Panel active vertical scanning start point and end point
VOP_DSP_VS_ST_END_F1	0x0198	W	0x00000000	Vertical scanning start point and vsync pulse end point of even filed in interlace mode
VOP_DSP_VACT_ST_END_F1	0x019c	W	0x00000000	Vertical scanning active start point and end point of even filed in interlace mode
VOP_PWM_CTRL	0x01a0	W	0x0000200a	PWM Control Register
VOP_PWM_PERIOD_HPR	0x01a4	W	0x00000000	PWM Period Register/High Polarity Capture Register
VOP_PWM_DUTY_LPR	0x01a8	W	0x00000000	PWM Duty Register/Low Polarity Capture Register
VOP_PWM_CNT	0x01ac	W	0x00000000	PWM Counter Register

Name	Offset	Size	Reset Value	Description
VOP_BCSH_COLOR_BAR	0x01b0	W	0x00000000	color bar config register
VOP_BCSH_BCS	0x01b4	W	0xd0010000	brightness contrast saturation*contrast config register
VOP_BCSH_H	0x01b8	W	0x01000000	sin hue and cos hue config register
VOP_BCSH_RESERVED	0x01bc	W	0x00000000	
VOP_CABC_CTRL0	0x01c0	W	0x00000000	
VOP_CABC_CTRL1	0x01c4	W	0x00000000	
VOP_CABC_GAUSS_LIN E0_0	0x01c8	W	0x15110903	Register0000 Abstract
VOP_CABC_GAUSS_LIN E0_1	0x01cc	W	0x00030911	Register0001 Abstract
VOP_CABC_GAUSS_LIN E1_0	0x01d0	W	0x1a150b04	Register0002 Abstract
VOP_CABC_GAUSS_LIN E1_1	0x01d4	W	0x00040b15	Register0003 Abstract
VOP_CABC_GAUSS_LIN E2_0	0x01d8	W	0x15110903	Register0004 Abstract
VOP_CABC_GAUSS_LIN E2_1	0x01dc	W	0x00030911	Register0005 Abstract
VOP_FRC_LOWER01_0	0x01e0	W	0x12844821	
VOP_FRC_LOWER01_1	0x01e4	W	0x21488412	
VOP_FRC_LOWER10_0	0x01e8	W	0xa55a9696	
VOP_FRC_LOWER10_1	0x01ec	W	0x5aa56969	
VOP_FRC_LOWER11_0	0x01f0	W	0xdeb77bed	
VOP_FRC_LOWER11_1	0x01f4	W	0xed7bb7de	
VOP_FRC_RESERVED0	0x01f8	W	0x00000000	
VOP_FRC_RESERVED1	0x01fc	W	0x00000000	
VOP_MMU_DTE_ADDR	0x0300	W	0x00000000	MMU current page Table address
VOP_MMU_STATUS	0x0304	W	0x00000000	MMU status register
VOP_MMU_COMMAND	0x0308	W	0x00000000	MMU command register
VOP_MMU_PAGE_FAULT _ADDR	0x030c	W	0x00000000	MMU logical address of last page fault
VOP_MMU_ZAP_ONE_LINE	0x0310	W	0x00000000	MMU Zap cache line register
VOP_MMU_INT_RAWSTA T	0x0314	W	0x00000000	MMU raw interrupt status register
VOP_MMU_INT_CLEAR	0x0318	W	0x00000000	MMU raw interrupt status register

Name	Offset	Size	Reset Value	Description
VOP_MMU_INT_MASK	0x031c	W	0x00000000	MMU raw interrupt status register
VOP_MMU_INT_STATUS	0x0320	W	0x00000000	MMU raw interrupt status register
VOP_MMU_AUTO_GATING	0x0324	W	0x00000000	mmu auto gating
VOP_WIN2_LUT_ADDR	0x0400	W	0x00000000	
VOP_WIN3_LUT_ADDR	0x0800	W	0x00000000	
VOP_HWC_LUT_ADDR	0x0c00	W	0x00000000	
VOP_GAMMA_LUT_ADDR	0x1000	W	0x00000000	
VOP_MCU_BYPASS_WPORT	0x2200	W	0x00000000	Register0000 Abstract
VOP_MCU_BYPASS_RPORT	0x2300	W	0x00000000	Register0001 Abstract

Notes: *Size* : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

## 27.4.2 Detail Register Description

### VOP\_REG\_CFG\_DONE

Address: Operational Base + offset (0x0000)

Register config done flag

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	WO	0x0	reg_load_en lcdc register config done flag In the first setting of the register, the new value was saved into the mirror register. When all the register config finish, writing this register to enable the copyright of the mirror register to real register. Then register would be updated at the start of every frame.

### VOP\_VERSION\_INFO

Address: Operational Base + offset (0x0004)

Bit	Attr	Reset Value	Description
31:16	RW	0x0000	fpga_version
15:0	RW	0x0000	rtl_version

### VOP\_SYS\_CTRL

Address: Operational Base + offset (0x0008)

System control register0

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23	RW	0x1	<p>auto_gating_en LCDC layer axi-clk auto gating enable 1'b0 : disable auto gating 1'b1 : enable auto gating default auto gating enable</p>
22	RW	0x0	<p>vop_standby_en LCDC standby mode Writing "1" to turn LCDC into standby mode, All the layer would disable and the data transfer from frame buffer memory would stop at the end of current frame. The output would be blank. When writing "0" to this bit, standby mode would disable and the LCDC go back to work immediately. 1'b0 : disable 1'b1 : enable * Black display is recommended before setting standby mode enable.</p>
21	RW	0x0	<p>vop_dma_stop LCDC DMA stop mode 1'b0 : disable 1'b1 : enable * If DMA is working, the stop mode would not be active until current bus transfer is finished.</p>
20	RW	0x0	<p>vop_mmu_en vop mmu enable signal 1'b0 : bypass mmu 1'b1 : enable mmu</p>
19:18	RW	0x0	<p>dma_burst_length DMA read Burst length 2'b00 : burst16 (burst 15 in rgb888 pack mode) 2'b01 : burst8 (burst 12 in rgb888 pack mode) 2'b10 : burst4 (burst 6 in rgb888 pack mode)</p>
17:16	RO	0x0	reserved
15	RW	0x0	<p>mipi_out_en 1'b0 : gating output clk ,data and control signal 1'b1 : mipi interface enable</p>
14	RW	0x0	<p>edp_out_en 1'b0 : gating output clk ,data and control signal 1'b1 : edp interface enable</p>
13	RW	0x0	<p>hdmi_out_en 1'b0 : gating output clk ,data and control signal 1'b1 : hdmi interface enable</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x1	rgb_out_en 1'b0 : gating output clk ,data and control signal 1'b1 : rgb/lvds interface enable
11	RO	0x0	reserved
10	RW	0x0	edpi_wms_fs edpi wms mode , rame st signal  write 褪: edpi_wms_mode frame start (when other register is config done) read : wms mode hold status
9	RW	0x0	edpi_wms_mode 1'b1: mipi command mode
8	RW	0x0	edpi_halt_en mipi flow ctrl enable
7:4	RW	0x0	doub_ch_overlap_num 4'h0: overlap num 0 4'h1: overlap num 2 4'h2: overlap num 4 4'h3: overlap num 6 4'h4: overlap num 8 4'h5: overlap num 10 4'h6: overlap num 12 4'h7: overlap num 14 4'h8: overlap num 16
3	RW	0x0	doub_channel_en mipi double channel enable
2:1	RW	0x0	direct_path_layer_sel direct path layer select 2'b00 : select win0 2'b01 : select win1 2'b10 : select win2 2'b11 : select win3
0	RW	0x0	direct_path_en iep direct path enable signal 1'b0 : disable iep direct path 1'b1 : enable iep direct path

**VOP\_SYS\_CTRL1**

Address: Operational Base + offset (0x000c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:18	RO	0x0	reserved
17:13	RW	0x00	axi_outstanding_max_num

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x0	axi_max_outstanding_en
11:10	RW	0x0	noc_win_qos
9	RW	0x0	noc_qos_en
8:3	RW	0x00	noc_hurry_threshold
2:1	RW	0x0	noc_hurry_value
0	RW	0x0	noc_hurry_en

**VOP\_DSP\_CTRL0**

Address: Operational Base + offset (0x0010)

Display control register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23	RW	0x0	dsp_y_mir_en 1'b0 : no y_mirror 1'b1 : y_mirror
22	RW	0x0	dsp_x_mir_en 1'b0 : no x_mirror 1'b1 : x_mirror
21	RW	0x0	dsp_yuv_clip YCrCb clip 1'b0 : disable, YCbCr no clip 1'b1 : enable, YCbCr clip before YCbCr2RGB10bit *Y clip: 64~940, CbCr clip: 64~960
20	RW	0x0	dsp_ccir656_avg Cb-Cr filter in CCIR656 mode 1'b0 : drop mode 1'b1 : average mode
19	RW	0x0	dsp_black_en Black display mode When this bit enable, the pixel data output is all black (0x000000)
18	RW	0x0	dsp_blank_en Blank display mode When this bit enable, the Hsync/Vsync/Den output is blank

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17	RW	0x0	dsp_out_zero Hsync/Vsync/Den output software ctrl 1'b0 : normal output 1'b1 : all output '0'
16	RW	0x0	dsp_dummy_swap Display dummy swap enable 1'b0 : B+G+R+dummy 1'b1 : dummy+B+G+R
15	RW	0x0	dsp_delta_swap Display delta swap enable 1'b0 : disable 1'b1 : enable *See detail description in Delta display chapter.
14	RW	0x0	dsp_rg_swap Display output red and green swap enable 1'b0 : RGB 1'b1 : GRB
13	RW	0x0	dsp_rb_swap Display output red and blue swap enable 1'b0 : RGB 1'b1 : BGR
12	RW	0x0	dsp_bg_swap Display output blue and green swap enable 1'b0 : RGB 1'b1 : RBG
11	RW	0x0	dsp_field_pol field polarity when interlace dsp 1'b0 : normal 1'b1 : invert
10	RW	0x0	dsp_interlace Interlace display enable 1'b0 : disable 1'b1 : enable *This mode is related to the ITU-R656 output, the display timing of odd field must be set correctly. (lcdc_dsp_vs_st_end_f1/lcdc_dsp_vact_end_f1)
9	RW	0x0	dsp_ddr_phase dclk phase lock 1'b0 : no lock 1'b1 : lock every line
8	RW	0x0	dsp_dclk_ddr dclk output mode 1'b0 : SDR 1'b1 : DDR

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	dsp_dclk_pol DCLK invert enable 1'b0 : normal 1'b1 : invert default dclk invert
6	RW	0x0	dsp_den_pol DEN polarity 1'b0 : positive 1'b1 : negative
5	RW	0x0	dsp_vsync_pol VSYNC polarity 1'b0 : negative 1'b1 : positive
4	RW	0x0	dsp_hsync_pol HSYNC polarity 1'b0 : negative 1'b1 : positive
3:0	RW	0x0	dsp_out_mode Display output format 4'b0000: Parallel 24-bit RGB888 output R[7:0],G[7:0],B[7:0] 4'b0001: Parallel 18-bit RGB666 output 6'b0,R[5:0],G[5:0],B[5:0] 4'b0010: Parallel 16-bit RGB565 output 8'b0,R[4:0],G[5:0],B[4:0] 4'b0011: Parallel 24-bit RGB888 double pixel mix out phase0:G1[3:0],B1[7:0],G0[3:0],B0[7:0] phase1:R1[7:0],G1[7:4],R0[7:0],G0[7:4] 4'b0100: Serial 2x12-bit 12'b0,G[3:0],B[7:0] + 12'b0,R[7:0],G[7:4] 4'b0101: ITU-656 output mode0 16'b0,pixel_data[7:0] 4'b0110: ITU-656 output mode1 8'b0,pixel_data[7:0],8'b0 4'b0111: ITU-656 output mode2 9'b0,pixel_data[7:0],7'b0 4'b1000: Serial 3x8-bit RGB888 16'b0, B[7:0]+16'b0,G[7:0]+16'b0,R[7:0] 4'b1100: Serial 3x8-bit RGB888 + dummy 16'b0, B[7:0]+16'b0,G[7:0]+16'b0,R[7:0] + dummy 4'b1111: Parallel 30-bit RGBaaa output R[9:0],G[9:0],B[9:0] Others: Reserved.

**VOP\_DSP\_CTRL1**

Address: Operational Base + offset (0x0014)

Display control register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:14	RW	0x3	dsp_layer3_sel
13:12	RW	0x2	dsp_layer2_sel
11:10	RW	0x1	dsp_layer1_sel
9:8	RW	0x0	dsp_layer0_sel
7	RO	0x0	reserved
6	RW	0x0	dither_up_en
5	RO	0x0	reserved
4	RW	0x0	dither_down_sel dither down mode select 2'b0 : allegro 2'b1 : FRC
3	RW	0x0	dither_down_mode Dither-down mode 1'b0 : RGB888 to RGB565 1'b1 : RGB888 to RGB666
2	RW	0x0	dither_down_en Dither-down enable 1'b0 : disable 1'b1 : enable
1	RW	0x0	pre_dither_down_en 10bit -> 8bit (allegro)
0	RW	0x0	dsp_lut_en Display LUT ram enable 1'b0 : disable 1'b1 : enable *This bit should be "0" when CPU updates the LUT, and should be "1" when Display LUT mode enable.

**VOP\_DSP\_BG**

Address: Operational Base + offset (0x0018)

background color

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
29:20	RW	0x000	dsp_bg_red Background Red color
19:10	RW	0x000	dsp_bg_green Background Green color
9:0	RW	0x000	dsp_bg_blue Background Blue color

**VOP MCU CTRL**

Address: Operational Base + offset (0x001c)

MCU mode control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	mcu_type MCU LCD output SELECT
30	RW	0x0	mcu_bypass MCU LCD BYPASS MODE Select
29	RW	0x0	mcu_rs MCU LCD RS Select
28	RW	0x0	muc_frame_st Write"1" : MCU HOLD Mode Frame Start Read : MCU/LCDC standby HOLD status
27	RW	0x0	mcu_hold_mode MCU HOLD Mode Select
26	RW	0x0	mcu_clk_sel MCU_CLK_SEL for MCU bypass 1'b1 : MCU BYPASS sync with DCLK 1'b0 : MCU BYPASS sync with HCLK
25:20	RW	0x07	mcu_rw_pend MCU_RW signal end point (0-63)
19:16	RW	0x1	mcu_rw_pst MCU_RW signal start point (0-15)
15:10	RW	0x07	mcu_cs_pend MCU_CS signal end point (0-63)
9:6	RW	0x0	mcu_cs_pst MCU_CS signal start point (0-15)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:0	RW	0x08	mcu_pix_total MCU LCD Interface writing period (1-63)

**VOP\_INTR\_CTRL0**

Address: Operational Base + offset (0x0020)

Interrupt ctrl register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:12	RW	0x0000	dsp_line_frag_num Line number of the Line flag interrupt The display line number when the flag interrupt occur, the range is (0~ DSP_VTOTAL-1).
11	WO	0x0	bus_error_intr_clr Bus error Interrupt clear (Auto clear)
10	WO	0x0	line_frag_intr_clr Line flag Interrupt clear (Auto clear)
9	WO	0x0	fs_intr_clr Frame start interrupt clear (Auto clear)
8	WO	0x0	dsp_hold_valid_intr_clr display hold valid interrupt clear (Auto clear)
7	RW	0x0	bus_error_intr_en Bus error interrupt enable 1'b0 : disable 1'b1 : enable
6	RW	0x0	line_frag_intr_en Line flag Interrupt enable 1'b0 : disable 1'b1 : enable
5	RW	0x0	fs_intr_en Frame start interrupt enable 1'b0 : disable 1'b1 : enable
4	RW	0x0	dsp_hold_valid_intr_en display hold valid interrupt enable 1'b0 : disable 1'b1 : enable
3	RO	0x0	bus_error_intr_sts Bus error Interrupt status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RO	0x0	line_frag_intr_sts Line flag Interrupt status
1	RO	0x0	fs_intr_sts Frame start interrupt status
0	RO	0x0	dsp_hold_valid_intr_sts display hold valid interrupt status

**VOP\_INTR\_CTRL1**

Address: Operational Base + offset (0x0024)

Interrupt ctrl register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:23	RO	0x0	reserved
22	RW	0x0	pwm_gen_intr_clr
21	RW	0x0	post_buf_empty_intr_clr post line buffer empty interrupt clear(auto clear)
20	RW	0x0	hwc_empty_intr_clr hwc data empty interrupt clear(auto clear)
19	RW	0x0	win3_empty_intr_clr win3 data empty interrupt clear(auto clear)
18	RW	0x0	win2_empty_intr_clr win2 data empty interrupt clear(auto clear)
17	RW	0x0	win1_empty_intr_clr win1 data empty interrupt clear(auto clear)
16	W1C	0x0	win0_empty_intr_clr win0 data empty interrupt clear(auto clear)
15	RO	0x0	reserved
14	RW	0x0	pwm_gen_intr_en
13	RW	0x0	post_buf_empty_intr_en post line buffer empty interrupt enable signal 1'b0 : disable 1'b1 : enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x0	hwc_empty_intr_en hwc data empty interrupt enable signal 1'b0 : disable 1'b1 : enable
11	RW	0x0	win3_empty_intr_en win3 data empty interrupt enable signal 1'b0 : disable 1'b1 : enable
10	RW	0x0	win2_empty_intr_en win2 data empty interrupt enable signal 1'b0 : disable 1'b1 : enable
9	RW	0x0	win1_empty_intr_en win1 data empty interrupt enable signal 1'b0 : disable 1'b1 : enable
8	RW	0x0	win0_empty_intr_en win0 data empty interrupt enable signal 1'b0 : disable 1'b1 : enable
7	RO	0x0	reserved
6	RW	0x0	pwm_gen_intr_sts pwm generated interrupt 0: Channel 0 Interrupt not generated 1: Channel 0 Interrupt generated
5	RW	0x0	post_buf_empty_intr_sts post buffer empty interrupt status
4	RW	0x0	hwc_empty_intr_sts hwc data empty interrupt status
3	RW	0x0	win3_empty_intr_sts win3 data empty interrupt status
2	RW	0x0	win2_empty_intr_sts win2 data empty interrupt status
1	RW	0x0	win1_empty_intr_sts win1 data empty interrupt status
0	RO	0x0	win0_empty_intr_sts win0 data empty interrupt status

**VOP\_INTR\_RESERVED0**

Address: Operational Base + offset (0x0028)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	reserved

### **VOP\_INTR\_RESERVED1**

Address: Operational Base + offset (0x002c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	reserved

### **VOP\_WIN0\_CTRL0**

Address: Operational Base + offset (0x0030)

win0 ctrl register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:21	RO	0x0	reserved
20	RW	0x0	win0_yuv_clip YCrCb clip 1'b0 : disable, YCbCr no clip 1'b1 : enable, YCbCr clip before YCbCr2RGB *Y clip: 16~235, CbCr clip: 16~239
19	RW	0x0	win0_cbr_deflick Win0 Cbr deflick mode 1'b0 : disable 1'b1 : enable
18	RW	0x0	win0_yrgb_deflick win0 YRGB deflick mode 1'b0 : disable 1'b1 : enable
17	RO	0x0	reserved
16	RW	0x0	win0_ppas_zero_en 0:per_pix_alpha+scale,pix not change; 1:per_pix_alpha_scale,pix=0 when alpha=0;
15	RW	0x0	win0_uv_swap Win0 CbCr swap 1'b0 : CrCb 1'b1 : CbCr
14	RW	0x0	win0_mid_swap Win0 Y middle swap 1'b0 : Y3Y2Y1Y0 1'b1 : Y3Y1Y2Y0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13	RW	0x0	win0_alpha_swap win0 alpha swap 1'b0 : ARGB 1'b1 : RGBA
12	RW	0x0	win0_rb_swap win0 RGB RED and BLUE swap 1'b0 : RGB 1'b1 : BGR
11:10	RW	0x0	win0_csc_mode Win0 YUV2RGB Color space conversion: 2'b00/01 : mpeg 2'b10 : jpeg 2'b11 : hd
9	RW	0x0	win0_no_outstanding win0 AXI master read outstanding 1'b0 : enable 1'b1 : disable
8	RW	0x0	win0_interlace_read Win0 interlace read mode 1'b0 : disable 1'b1 : enable
7:5	RW	0x2	win0_lb_mode
4	RW	0x0	win0_fmt_10 0: yuv 8bit fmt mode 1: yuv 10bit fmt mode
3:1	RW	0x0	win0_data_fmt 3'b000 : ARGB888 3'b001 : RGB888 3'b010 : RGB565 3'b100 : YcbCr420 3'b101 : YcbCr422 3'b110 : YcbCr444
0	RW	0x0	win0_en

**VOP\_WIN0\_CTRL1**

Address: Operational Base + offset (0x0034)

win1 ctrl register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	win0_cbr_vsd_mode win0 vertical scaler down mode select 1'b0 : bilinear 1'b1 : average
30	RW	0x0	win0_cbr_vsu_mode win0 vertical scaler down mode select 1'b0 : bilinear 1'b1 : bicubic
29:28	RW	0x0	win0_cbr_hsd_mode win0 horizontal scaler down mode select 2'b00 : bilinear 2'b01 : bicubic 2'b10 : average
27:26	RW	0x0	win0_cbr_ver_scl_mode
25:24	RW	0x0	win0_cbr_hor_scl_mode 2'b00 : no scale 2'b01 : scale up 2'b10 : scale down 2'b11 : no scale
23	RW	0x0	win0_yrgb_vsd_mode win0 vertical scaler down mode select 1'b0 : bilinear 1'b1 : average
22	RW	0x0	win0_yrgb_vsu_mode win0 vertical scaler down mode select 1'b0 : bilinear 1'b1 : bicubic
21:20	RW	0x0	win0_yrgb_hsd_mode win0 horizontal scaler down mode select 2'b00 : bilinear 2'b01 : average
19:18	RW	0x0	win0_yrgb_ver_scl_mode 2'b00 : no scale 2'b01 : scale up 2'b10 : scale down 2'b11 : no scale
17:16	RW	0x0	win0_yrgb_hor_scl_mode 2'b00 : no scale 2'b01 : scale up 2'b10 : scale down 2'b11 : no scale

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RW	0x0	win0_line_load_mode when yuv fmt,if 1'b0: load data by axi trans 1'b1: load data by lines
14:12	RW	0x0	win0_cbr_axi_gather_num
11:8	RW	0x0	win0_yrgb_axi_gather_num
7	RW	0x0	win0_vsd_cbr_gt2
6	RW	0x0	win0_vsd_cbr_gt4
5	RW	0x0	win0_vsd_yrgb_gt2
4	RW	0x0	win0_vsd_yrgb_gt4
3:2	RW	0x0	win0_bic_coe_sel 2'b00 : PRECISE 2'b01 : SPLINE 2'b10 : CATROM 2'b11 : MITCHELL
1	RW	0x0	win0_cbr_axi_gather_en
0	RW	0x0	win0_yrgb_axi_gather_en

**VOP\_WIN0\_COLOR\_KEY**

Address: Operational Base + offset (0x0038)

Win0 color key register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	win0_key_en Win0 transparency color key enable 1'b0 : disable; 1'b1 : enable;
30	RO	0x0	reserved
29:0	RW	0x00000000	win0_key_color Win0 key color

**VOP\_WIN0\_VIR**

Address: Operational Base + offset (0x003c)

Win0 virtual stride

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
29:16	RW	0x0000	win0_vir_stride_uv
15:14	RO	0x0	reserved
13:0	RW	0x0140	win0_vir_stride Win0 Virtual stride Number of words of Win0 Virtual width ARGB888 : win0_vir_width RGB888 : (win0_vir_width*3/4) + (win0_vir_width%3) RGB565 : ceil(win0_vir_width/2) YUV : ceil(win0_vir_width/4)

**VOP\_WIN0\_YRGB\_MST**

Address: Operational Base + offset (0x0040)

Win0 YRGB memory start address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win0_yrgb_mst win0 YRGB frame buffer memory start address

**VOP\_WIN0\_CBR\_MST**

Address: Operational Base + offset (0x0044)

Win0 Cbr memory start address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win0_cbr_mst win0 CBR frame buffer memory start address

**VOP\_WIN0\_ACT\_INFO**

Address: Operational Base + offset (0x0048)

Win0 active window width/height

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x00ef	win0_act_height Win0 active(original) window height win_act_height = (win0 vertical size -1)
15:13	RO	0x0	reserved
12:0	RW	0x013f	win0_act_width Win0 active(original) window width win_act_width = (win0 horizontal size -1)

**VOP\_WIN0\_DSP\_INFO**

Address: Operational Base + offset (0x004c)

Win0 display width/height on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:16	RW	0x0ef	win0_dsp_height Win0 display window height win0_dsp_height = (win0 vertical size -1)
15:12	RO	0x0	reserved
11:0	RW	0x13f	win0_dsp_width Win0 display window width win0_dsp_width = (win0 horizontal size -1)

**VOP\_WIN0\_DSP\_ST**

Address: Operational Base + offset (0x0050)

Win0 display start point on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	win0_dsp_yst Win0 vertical start point(y) of the Panel scanning
15:13	RO	0x0	reserved
12:0	RW	0x000a	win0_dsp_xst Win0 horizontal start point(x) of the Panel scanning

**VOP\_WIN0\_SCL\_FACTOR\_YRGB**

Address: Operational Base + offset (0x0054)

Win0 YRGB scaling factor

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x1000	win0_vs_factor_yrgb Win0 YRGB vertical scaling factor: factor=((LCDC_WIN0_ACT_INFO[31:16]) /(LCDC_WIN0_DSP_INFO[31:16]))*2^12
15:0	RW	0x1000	win0_hs_factor_yrgb Win0 YRGB horizontal scaling factor: factor=((LCDC_WIN0_ACT_INFO[15:0]) /(LCDC_WIN0_DSP_INFO[15:0]))*2^12

**VOP\_WIN0\_SCL\_FACTOR\_CBR**

Address: Operational Base + offset (0x0058)

Win0 Cbr scaling factor

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x1000	win0_vs_factor_cbr Win0 CBR vertical scaling factor: YCbCr420: factor=((LCDC_WIN0_ACT_INFO[31:16]/ 2) /(LCDC_WIN0_DSP_INFO[31:16] ))*2^12 YCbCr422,YCbCr444: factor=((LCDC_WIN0_ACT_INFO[31:16]) /(LCDC_WIN0_DSP_INFO[31:16] ))*2^12
15:0	RW	0x1000	win0_hs_factor_cbr Win0 CBR horizontal scaling factor: YCbCr422,YCbCr420: factor=((LCDC_WIN0_ACT_INFO[15:0]/2) /(LCDC_WIN0_DSP_INFO[15:0]))*2^12 YCbCr444: factor=((LCDC_WIN0_ACT_INFO[15:0]) /(LCDC_WIN0_DSP_INFO[15:0]))*2^12

**VOP\_WIN0\_SCL\_OFFSET**

Address: Operational Base + offset (0x005c)

Win0 scaling start point offset

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	win0_vs_offset_cbr Cbr Vertical scaling start point offset (0x00~0xff)/0x100 = 0~0.99
23:16	RW	0x00	win0_vs_offset_yrgb Y Vertical scaling start point offset (0x00~0xff)/0x100 = 0~0.99
15:8	RW	0x00	win0_hs_offset_cbr Cbr Horizontal scaling start point offset (0x00~0xff)/0x100 = 0~0.99
7:0	RW	0x00	win0_hs_offset_yrgb Y Horizontal scaling start point offset (0x00~0xff)/0x100 = 0~0.99

**VOP\_WIN0\_SRC\_ALPHA\_CTRL**

Address: Operational Base + offset (0x0060)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	win0_fading_value
23:16	RW	0x00	win0_src_global_alpha layer0 src global alpha (eused by fading value)
15:9	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8:6	RW	0x0	win0_src_factor_m0
5	RW	0x0	win0_src_alpha_cal_m0
4:3	RW	0x0	win0_src_blend_m0
2	RW	0x0	win0_src_alpha_m0
1	RW	0x0	win0_src_color_m0
0	RW	0x0	win0_src_alpha_en

**VOP\_WIN0\_DST\_ALPHA\_CTRL**

Address: Operational Base + offset (0x0064)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8:6	RW	0x0	win0_dst_factor_m0
5:0	RW	0x00	win0_dst_m0_reserved

**VOP\_WIN0\_FADEING\_CTRL**

Address: Operational Base + offset (0x0068)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	layer0_fading_en
23:16	RW	0x00	layer0_fading_offset_b
15:8	RW	0x00	layer0_fading_offset_g
7:0	RW	0x00	layer0_fading_offset_r

**VOP\_WIN0\_RESERVED0**

Address: Operational Base + offset (0x006c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	reserved

**VOP\_WIN1\_CTRL0**

Address: Operational Base + offset (0x0070)

win1 ctrl register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:21	RO	0x0	reserved
20	RW	0x0	win1_yuv_clip YCrCb clip 1'b0 : disable, YCbCr no clip 1'b1 : enable, YCbCr clip before YCbCr2RGB *Y clip: 16~235, CbCr clip: 16~239
19	RW	0x0	win1_cbr_deflick Win1 Cbr deflick mode 1'b0 : disable 1'b1 : enable
18	RW	0x0	win1_yrgb_deflick win1 YRGB deflick mode 1'b0 : disable 1'b1 : enable
17	RO	0x0	reserved
16	RW	0x0	win1_ppas_zero_en 0:per_pix_alpha+scale,pix not change; 1:per_pix_alpha_scale,pix=0 when alpha=0;
15	RW	0x0	win1_uv_swap Win1 CbCr swap 1'b0 : CrCb 1'b1 : CbCr
14	RW	0x0	win1_mid_swap Win1 Y middle 8-bit swap 1'b0 : Y3Y2Y1Y0 1'b1 : Y3Y1Y2Y0
13	RW	0x0	win1_alpha_swap win1 alpha swap 1'b0 : ARGB 1'b1 : RGBA
12	RW	0x0	win1_rb_swap win1 RGB RED and BLUE swap 1'b0 : RGB 1'b1 : BGR
11:10	RW	0x0	win1_csc_mode Win1 YUV2RGB Color space conversion: 2'b00/01 : mpeg 2'b10 : jpeg 2'b11 : hd

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	win1_no_outstanding win1 AXI master read outstanding 1'b0 : enable 1'b1 : disable
8	RW	0x0	win1_interlace_read Win1 interlace read mode 1'b0 : disable 1'b1 : enable
7:5	RW	0x2	win1_lb_mode
4	RW	0x0	win1_fmt_10 0: yuv 8bit fmt mode 1: yuv 10bit fmt mode
3:1	RW	0x0	win1_data_fmt 3'b000 : ARGB888 3'b001 : RGB888 3'b010 : RGB565 3'b100 : YcbCr420 3'b101 : YcbCr422 3'b110 : YcbCr444
0	RW	0x0	win1_en

**VOP\_WIN1\_CTRL1**

Address: Operational Base + offset (0x0074)

win1 ctrl register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	win1_cbr_vsd_mode win1 vertical scaler down mode select 1'b0 : bilinear 1'b1 : average
30	RW	0x0	win1_cbr_vsu_mode win1 vertical scaler up mode select 1'b0 : bilinear 1'b1 : bicubic
29:28	RW	0x0	win1_cbr_hsd_mode win1 horizontal scaler down mode select 2'b00 : bilinear 2'b01 : bicubic 2'b10 : average

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27:26	RW	0x0	win1_cbr_ver_scl_mode 2'b00 : no scale 2'b01 : scale up 2'b10 : scale down 2'b11 : no scale
25:24	RW	0x0	win1_cbr_hor_scl_mode 2'b00 : no scale 2'b01 : scale up 2'b10 : scale down 2'b11 : no scale
23	RW	0x0	win1_yrgb_vsd_mode win1 vertical scaler down mode select 1'b0 : bilinear 1'b1 : average
22	RW	0x0	win1_yrgb_vsu_mode win1 vertical scaler up mode select 1'b0 : bilinear 1'b1 : bicubic
21:20	RW	0x0	win1_yrgb_hsd_mode win1 horizontal scaler down mode select 2'b00 : bilinear 2'b01 : average
19:18	RW	0x0	win1_yrgb_ver_scl_mode 2'b00 : no scale 2'b01 : scale up 2'b10 : scale down 2'b11 : no scale
17:16	RW	0x0	win1_yrgb_hor_scl_mode 2'b00 : no scale 2'b01 : scale up 2'b10 : scale down 2'b11 : no scale
15	RW	0x0	win1_line_load_mode when yuv fmt,if 1'b0: load data by pixels 1'b1: load data by lines
14:12	RW	0x0	win1_cbr_axi_gather_num
11:8	RW	0x0	win1_yrgb_axi_gather_num
7	RW	0x0	win1_vsd_cbr_gt2
6	RW	0x0	win1_vsd_cbr_gt4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	win1_vsd_yrgb_gt2
4	RW	0x0	win1_vsd_yrgb_gt4
3:2	RW	0x0	win1_bic_coe_sel 2'b00 : PRECISE 2'b01 : SPLINE 2'b10 : CATROM 2'b11 : MITCHELL
1	RW	0x0	win1_cbr_axi_gather_en
0	RW	0x0	win1_yrgb_axi_gather_en

**VOP\_WIN1\_COLOR\_KEY**

Address: Operational Base + offset (0x0078)

Win1 color key register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	win1_key_en Win1 transparency color key enable 1'b0 : disable; 1'b1 : enable;
30	RO	0x0	reserved
29:0	RW	0x00000000	win1_key_color Win1 key color

**VOP\_WIN1\_VIR**

Address: Operational Base + offset (0x007c)

win1 virtual stride

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved
29:16	RW	0x0000	win1_vir_stride_uv
15:14	RO	0x0	reserved
13:0	RW	0x0140	win1_vir_stride Win1 Virtual stride Number of words of Win1 Virtual width ARGB888 : win1_vir_width RGB888 : (win1_vir_width*3/4) + (win1_vir_width%3) RGB565 : ceil(win1_vir_width/2) YUV : ceil(win1_vir_width/4)

**VOP\_WIN1\_YRGB\_MST**

Address: Operational Base + offset (0x0080)

Win1 YRGB memory start address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win1_yrgb_mst win1 YRGB frame buffer memory start address

**VOP\_WIN1\_CBR\_MST**

Address: Operational Base + offset (0x0084)

Win1 Cbr memory start address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win1_cbr_mst win1 CBR frame buffer memory start address

**VOP\_WIN1\_ACT\_INFO**

Address: Operational Base + offset (0x0088)

Win1 active window width/height

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x00ef	win1_act_height Win1 active(original) window height win_act_height = (win1 vertical size -1)
15:13	RO	0x0	reserved
12:0	RW	0x013f	win1_act_width Win1 active(original) window width win_act_width = (win1 horizontal size -1)

**VOP\_WIN1\_DSP\_INFO**

Address: Operational Base + offset (0x008c)

Win1 display width/height on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:16	RW	0x0ef	win1_dsp_height Win1 display window height win1_dsp_height = (win1 vertical size -1)
15:12	RO	0x0	reserved
11:0	RW	0x13f	win1_dsp_width Win1 display window width win1_dsp_width = (win1 horizontal size -1)

**VOP\_WIN1\_DSP\_ST**

Address: Operational Base + offset (0x0090)

Win1 display start point on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	win1_dsp_yst Win1 vertical start point(y) of the Panel scanning
15:13	RO	0x0	reserved
12:0	RW	0x000a	win1_dsp_xst Win1 horizontal start point(x) of the Panel scanning

**VOP\_WIN1\_SCL\_FACTOR\_YRGB**

Address: Operational Base + offset (0x0094)

Win1 YRGB scaling factor

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x1000	win1_vs_factor_yrgb Win1 YRGB vertical scaling factor: factor=((LCDC_WIN1_ACT_INFO[31:16]) /(LCDC_WIN1_DSP_INFO[31:16]))*2^12
15:0	RW	0x1000	win1_hs_factor_yrgb Win1 YRGB horizontal scaling factor: factor=((LCDC_WIN1_ACT_INFO[15:0]) /(LCDC_WIN1_DSP_INFO[15:0]))*2^12

**VOP\_WIN1\_SCL\_FACTOR\_CBR**

Address: Operational Base + offset (0x0098)

Win1 Cbr scaling factor

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x1000	win1_vs_factor_cbr Win1 CBR vertical scaling factor: YCbCr420: factor=((LCDC_WIN1_ACT_INFO[31:16]/ 2) /(LCDC_WIN1_DSP_INFO[31:16] ))*2^12 YCbCr422,YCbCr444: factor=((LCDC_WIN1_ACT_INFO[31:16]) /(LCDC_WIN1_DSP_INFO[31:16] ))*2^12
15:0	RW	0x1000	win1_hs_factor_cbr Win1 Cbr horizontal scaling factor: YCbCr422,YCbCr420: factor=((LCDC_WIN1_ACT_INFO[15:0]/2) /(LCDC_WIN1_DSP_INFO[15:0]))*2^12 YCbCr444: factor=((LCDC_WIN1_ACT_INFO[15:0]) /(LCDC_WIN1_DSP_INFO[15:0]))*2^12

**VOP\_WIN1\_SCL\_OFFSET**

Address: Operational Base + offset (0x009c)

Win1 scaling start point offset

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	win1_vs_offset_cbr Cbr Vertical scaling start point offset (0x00~0xff)/0x100 = 0~0.99
23:16	RW	0x00	win1_vs_offset_yrgb Y Vertical scaling start point offset (0x00~0xff)/0x100 = 0~0.99
15:8	RW	0x00	win1_hs_offset_cbr Cbr Horizontal scaling start point offset (0x00~0xff)/0x100 = 0~0.99
7:0	RW	0x00	win1_hs_offset_yrgb Y Horizontal scaling start point offset (0x00~0xff)/0x100 = 0~0.99

**VOP\_WIN1\_SRC\_ALPHA\_CTRL**

Address: Operational Base + offset (0x00a0)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	win1_fading_value
23:16	RW	0x00	win1_src_global_alpha layer0 src global alpha (eused by fading value)
15:9	RO	0x0	reserved
8:6	RW	0x0	win1_src_factor_m0
5	RW	0x0	win1_src_alpha_cal_m0
4:3	RW	0x0	win1_src_blend_m0
2	RW	0x0	win1_src_alpha_m0
1	RW	0x0	win1_src_color_m0
0	RW	0x0	win1_src_alpha_en

**VOP\_WIN1\_DST\_ALPHA\_CTRL**

Address: Operational Base + offset (0x00a4)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8:6	RW	0x0	win1_dst_factor_m0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:0	RW	0x00	win1_dsp_m0_reserved

**VOP\_WIN1\_FADING\_CTRL**

Address: Operational Base + offset (0x00a8)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	win1_fading_en
23:16	RW	0x00	win1_fading_offset_b
15:8	RW	0x00	win1_fading_offset_g
7:0	RW	0x00	win1_fading_offset_r

**VOP\_WIN1\_RESERVED0**

Address: Operational Base + offset (0x00ac)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	reserved

**VOP\_WIN2\_CTRL0**

Address: Operational Base + offset (0x00b0)

win2 ctrl register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:19	RO	0x0	reserved
18	RW	0x0	win2_lut_en Win2 LUT ram enable 1'b0 : disable 1'b1 : enable *This bit should be "0" when CPU updates the LUT, and should be "1" when Win1 LUT mode enable.
17:15	RO	0x0	reserved
14	RW	0x0	win2_endian_swap Win2 8pp palette data Big-endian/ Little-endian select 1'b0 : Big-endian 1'b1 : Little-endian
13	RW	0x0	win2_alpha_swap Win2 RGB alpha swap 1'b0 : ARGB 1'b1 : RGBA

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RW	0x0	win2_rb_swap Win2 RGB Red and Blue swap 1'b0 : RGB 1'b1 : BGR
11	RO	0x0	reserved
10	RW	0x0	win2_csc_mode Win2 RGB2YUV Color space conversion: 1'b0 : no CSC 1'b1 : RGB2YUV
9	RW	0x0	win2_no_outstanding Win2 AXI master read outstanding 1'b0 : enable 1'b1 : disable
8	RW	0x0	win2_interlace_read Win2 interlace read mode 1'b0 : disable 1'b1 : enable
7	RW	0x0	win2_mst3_en win2 master3 enable
6	RW	0x0	win2_mst2_en win2 master2 enable
5	RW	0x0	win2_mst1_en win2 master1 enable
4	RW	0x0	win2_mst0_en win2 master0 enable
3:1	RW	0x0	win2_data_fmt 3'b000 : ARGB888 3'b001 : RGB888 3'b010 : RGB565 3'b100: 8bpp 3'b101: 4bpp 3'b110: 2bpp 3'b111: 1bpp
0	RW	0x0	win2_en

**VOP\_WIN2\_CTRL1**

Address: Operational Base + offset (0x00b4)  
win2 ctrl register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:4	RW	0x0	win2_axi_gather_num
3:1	RO	0x0	reserved
0	RW	0x0	win2_axi_gather_en

**VOP\_WIN2\_VIRO\_1**

Address: Operational Base + offset (0x00b8)

Win2 virtual stride0 and virtaul stride1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x0140	win2_vir_stride1 Win2 Virtual stride1 Number of words of Win2 Virtual1 width ARGB888 : win2_vir_width1 RGB888 : (win2_vir_width1 * 3/4) + (win2_vir_width1 % 3) RGB565 : ceil(win2_vir_width1 / 2) 8BPP : ceil(win2_vir_width1 / 4) 4BPP : ceil(win2_vir_width1 / 8) 2BPP : ceil(win2_vir_width1 / 16) 1BPP : ceil(win2_vir_width1 / 32)
15:13	RO	0x0	reserved
12:0	RW	0x0140	win2_vir_stride0 Win2 Virtual stride0 Number of words of Win2 Virtual0 width ARGB888 : win2_vir_width0 RGB888 : (win2_vir_width0 * 3/4) + (win2_vir_width0 % 3) RGB565 : ceil(win2_vir_width0 / 2) 8BPP : ceil(win2_vir_width0 / 4) 4BPP : ceil(win2_vir_width0 / 8) 2BPP : ceil(win2_vir_width0 / 16) 1BPP : ceil(win2_vir_width0 / 32)

**VOP\_WIN2\_VIR2\_3**

Address: Operational Base + offset (0x00bc)

Win2 virtual stride2 and virtaul stride3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
28:16	RW	0x0140	win2_vir_stride3 Win2 Virtual stride3 Number of words of Win2 Virtual3 width ARGB888 : win2_vir_width3 RGB888 : (win2_vir_width3 * 3/4) + (win2_vir_width3 % 3) RGB565 : ceil(win2_vir_width3 / 2) 8BPP : ceil(win2_vir_width3 / 4) 4BPP : ceil(win2_vir_width3 / 8) 2BPP : ceil(win2_vir_width3 / 16) 1BPP : ceil(win1_vir_width3 / 32)
15:13	RO	0x0	reserved
12:0	RW	0x0140	win2_vir_stride2 Win2 Virtual stride2 Number of words of Win2 Virtual2 width ARGB888 : win2_vir_width2 RGB888 : (win2_vir_width2 * 3/4) + (win2_vir_width2 % 3) RGB565 : ceil(win2_vir_width2 / 2) 8BPP : ceil(win2_vir_width2 / 4) 4BPP : ceil(win2_vir_width2 / 8) 2BPP : ceil(win2_vir_width2 / 16) 1BPP : ceil(win1_vir_width2 / 32)

**VOP\_WIN2\_MST0**

Address: Operational Base + offset (0x00c0)

Win2 memory start address0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win2_mst0 Win2 frame buffer memory start address0 *must be aliased to 8byte address

**VOP\_WIN2\_DSP\_INFO0**

Address: Operational Base + offset (0x00c4)

Win2 display width0/height0 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:16	RW	0x0ef	win2_dsp_height0 Win2 display window height0 win2_dsp_height0 = size -1
15:12	RO	0x0	reserved
11:0	RW	0x13f	win2_dsp_width0 Win2 display window width0 win2_dsp_width = size -1

**VOP\_WIN2\_DSP\_ST0**

Address: Operational Base + offset (0x00c8)

Win2 display start point0 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	win2_dsp_yst0 Win2 vertical start point(y) of the Panel scanning
15:13	RO	0x0	reserved
12:0	RW	0x000a	win2_dsp_xst0 Win2 horizontal start point(x) of the Panel scanning

**VOP\_WIN2\_COLOR\_KEY**

Address: Operational Base + offset (0x00cc)

Win2 color key register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	win2_key_en Win2 transparency color key enable 1'b0 : disable; 1'b1 : enable;
23:0	RW	0x0000000	win2_key_color Win2 key color

**VOP\_WIN2\_MST1**

Address: Operational Base + offset (0x00d0)

Win2 memory start address1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win2_mst1 Win2 frame buffer memory start address1 *must be aliased to 8byte address

**VOP\_WIN2\_DSP\_INFO1**

Address: Operational Base + offset (0x00d4)

Win2 display width1/height1 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:16	RW	0x0ef	win2_dsp_height1 Win2 display window height1 win2_dsp_height0 = size -1
15:12	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:0	RW	0x13f	win2_dsp_width1 Win2 display window width1 win2_dsp_width = size -1

**VOP\_WIN2\_DSP\_ST1**

Address: Operational Base + offset (0x00d8)

Win2 display start point1 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	win2_dsp_yst1 Win2 vertical start point(y) of the Panel scanning
15:13	RO	0x0	reserved
12:0	RW	0x000a	win2_dsp_xst1 Win2 horizontal start point(x) of the Panel scanning

**VOP\_WIN2\_SRC\_ALPHA\_CTRL**

Address: Operational Base + offset (0x00dc)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	win2_fading_value
23:16	RW	0x00	win2_src_global_alpha layer0 src global alpha (eused by fading value)
15:9	RO	0x0	reserved
8:6	RW	0x0	win2_src_factor_m0
5	RW	0x0	win2_src_alpha_cal_m0
4:3	RW	0x0	win2_src_blend_m0
2	RW	0x0	win2_src_alpha_m0
1	RW	0x0	win2_src_color_m0
0	RW	0x0	win2_src_alpha_en

**VOP\_WIN2\_MST2**

Address: Operational Base + offset (0x00e0)

Win2 memory start address2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win2_mst2 Win2 frame buffer memory start address2 *must be aliased to 8byte address

**VOP\_WIN2\_DSP\_INFO2**

Address: Operational Base + offset (0x00e4)

Win2 display width2/height2 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:16	RW	0x0ef	win2_dsp_height2 Win2 display window height2 win2_dsp_height0 = size -1
15:12	RO	0x0	reserved
11:0	RW	0x13f	win2_dsp_width2 Win2 display window width2 win2_dsp_width = size -1

**VOP\_WIN2\_DSP\_ST2**

Address: Operational Base + offset (0x00e8)

Win2 display start point2 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	win2_dsp_yst2 Win2 vertical start point(y) of the Panel scanning
15:13	RO	0x0	reserved
12:0	RW	0x000a	win2_dsp_xst2 Win2 horizontal start point(x) of the Panel scanning

**VOP\_WIN2\_DST\_ALPHA\_CTRL**

Address: Operational Base + offset (0x00ec)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8:6	RW	0x0	win2_dst_factor_m0
5:0	RW	0x00	win2_dst_m0_reserved

**VOP\_WIN2\_MST3**

Address: Operational Base + offset (0x00f0)

Win2 memory start address3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win2_mst3 Win2 frame buffer memory start address3 *must be aliased to 8byte address

**VOP\_WIN2\_DSP\_INFO3**

Address: Operational Base + offset (0x00f4)

Win2 display width3/height3 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:16	RW	0x0ef	win2_dsp_height3 Win2 display window height3 win2_dsp_height0 = size -1
15:12	RO	0x0	reserved
11:0	RW	0x13f	win2_dsp_width3 Win2 display window width3 win2_dsp_width = size -1

**VOP\_WIN2\_DSP\_ST3**

Address: Operational Base + offset (0x00f8)

Win2 display start point3 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	win2_dsp_yst3 Win2 vertical start point(y) of the Panel scanning
15:13	RO	0x0	reserved
12:0	RW	0x000a	win2_dsp_xst3 Win2 horizontal start point(x) of the Panel scanning

**VOP\_WIN2\_FADING\_CTRL**

Address: Operational Base + offset (0x00fc)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	win2_fading_en
23:16	RW	0x00	win2_fading_offset_b
15:8	RW	0x00	win2_fading_offset_g
7:0	RW	0x00	win2_fading_offset_r

**VOP\_WIN3\_CTRL0**

Address: Operational Base + offset (0x0100)

win0 ctrl register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:19	RO	0x0	reserved
18	RW	0x0	win3_lut_en Win3 LUT ram enable 1'b0 : disable 1'b1 : enable *This bit should be "0" when CPU updates the LUT, and should be "1" when Win1 LUT mode enable.
17:15	RO	0x0	reserved
14	RW	0x0	win3_endian_swap Win3 8pp palette data Big-endian/ Little-endian select 1'b0 : Big-endian 1'b1 : Little-endian
13	RW	0x0	win3_alpha_swap Win3 RGB alpha swap 1'b0 : ARGB 1'b1 : RGBA
12	RW	0x0	win3_rb_swap Win3 RGB Red and Blue swap 1'b0 : RGB 1'b1 : BGR
11	RO	0x0	reserved
10	RW	0x0	win3_csc_mode Win3 RGB2YUV Color space conversion: 1'b0 : no CSC 1'b1 : RGB2YUV
9	RW	0x0	win3_no_outstanding Win3 AXI master read outstanding 1'b0 : enable 1'b1 : disable
8	RW	0x0	win3_interlace_read Win3 interlace read mode 1'b0 : disable 1'b1 : enable
7	RW	0x0	win3_mst3_en win3 master3 enable
6	RW	0x0	win3_mst2_en win3 master2 enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	win3_mst1_en win3 master1 enable
4	RW	0x0	win3_mst0_en win3 master0 enable
3:1	RW	0x0	win3_data_fmt 3'b000 : ARGB888 3'b001 : RGB888 3'b010 : RGB565 3'b100: 8bpp 3'b101: 4bpp 3'b110: 2bpp 3'b111: 1bpp
0	RW	0x0	win3_en

**VOP\_WIN3\_CTRL1**

Address: Operational Base + offset (0x0104)

win0 ctrl register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:4	RW	0x0	win3_axi_gather_num
3:1	RO	0x0	reserved
0	RW	0x0	win3_axi_gather_en

**VOP\_WIN3\_VIRO\_1**

Address: Operational Base + offset (0x0108)

Win3 virtual stride0 and virtaul stride1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x0140	win3_vir_stride1 Win3 Virtual stride1 Number of words of Win3 Virtual1 width ARGB888 : win3_vir_width1 RGB888 : (win3_vir_width1 * 3/4) + (win3_vir_width1 % 3) RGB565 : ceil(win3_vir_width1 / 2) 8BPP : ceil(win3_vir_width1 / 4) 4BPP : ceil(win3_vir_width1 / 8) 2BPP : ceil(win3_vir_width1 / 16) 1BPP : ceil(win3_vir_width1 / 32)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:13	RO	0x0	reserved
12:0	RW	0x0140	win3_vir_stride0 Win3 Virtual stride0 Number of words of Win3 Virtual1 width ARGB888 : win3_vir_width1 RGB888 : (win3_vir_width1 * 3/4) + (win3_vir_width1 % 3) RGB565 : ceil(win3_vir_width1 / 2) 8BPP : ceil(win3_vir_width1 / 4) 4BPP : ceil(win3_vir_width1 / 8) 2BPP : ceil(win3_vir_width1 / 16) 1BPP : ceil(win3_vir_width1 / 32)

**VOP\_WIN3\_VIR2\_3**

Address: Operational Base + offset (0x010c)

Win3 virtual stride2 and virtaul stride3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x0140	win3_vir_stride3 Win3 Virtual stride3 Number of words of Win3 Virtual1 width ARGB888 : win3_vir_width1 RGB888 : (win3_vir_width1 * 3/4) + (win3_vir_width1 % 3) RGB565 : ceil(win3_vir_width1 / 2) 8BPP : ceil(win3_vir_width1 / 4) 4BPP : ceil(win3_vir_width1 / 8) 2BPP : ceil(win3_vir_width1 / 16) 1BPP : ceil(win3_vir_width1 / 32)
15:13	RO	0x0	reserved
12:0	RW	0x0140	win3_vir_stride2 Win3 Virtual stride2 Number of words of Win3 Virtual1 width ARGB888 : win3_vir_width1 RGB888 : (win3_vir_width1 * 3/4) + (win3_vir_width1 % 3) RGB565 : ceil(win3_vir_width1 / 2) 8BPP : ceil(win3_vir_width1 / 4) 4BPP : ceil(win3_vir_width1 / 8) 2BPP : ceil(win3_vir_width1 / 16) 1BPP : ceil(win3_vir_width1 / 32)

**VOP\_WIN3\_MST0**

Address: Operational Base + offset (0x0110)

Win3 memory start address0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win3_mst0 Win3 frame buffer memory start address0 *must be aliased to 8byte address

**VOP\_WIN3\_DSP\_INFO0**

Address: Operational Base + offset (0x0114)

Win3 display width0/height0 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:16	RW	0x0ef	win3_dsp_height0 Win3 display window height0 win3_dsp_height0 = size -1
15:12	RO	0x0	reserved
11:0	RW	0x13f	win3_dsp_width0 Win3 display window width0 win3_dsp_width = size -1

**VOP\_WIN3\_DSP\_ST0**

Address: Operational Base + offset (0x0118)

Win3 display start point0 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	win3_dsp_yst0 Win3 vertical start point(y) of the Panel scanning
15:13	RO	0x0	reserved
12:0	RW	0x000a	win3_dsp_xst0 Win3 horizontal start point(x) of the Panel scanning

**VOP\_WIN3\_COLOR\_KEY**

Address: Operational Base + offset (0x011c)

Win3 color key register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	win3_key_en Win3 transparency color key enable 1'b0 : disable; 1'b1 : enable;
23:0	RW	0x000000	win3_key_color Win3 key color

**VOP\_WIN3\_MST1**

Address: Operational Base + offset (0x0120)

Win3 memory start address1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win3_mst1 Win3 frame buffer memory start address1 *must be aliased to 8byte address

### VOP\_WIN3\_DSP\_INFO1

Address: Operational Base + offset (0x0124)

Win3 display width1/height1 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:16	RW	0x0ef	win3_dsp_height1 Win3 display window height1 win3_dsp_height0 = size -1
15:12	RO	0x0	reserved
11:0	RW	0x13f	win3_dsp_width1 Win3 display window width1 win3_dsp_width = size -1

### VOP\_WIN3\_DSP\_ST1

Address: Operational Base + offset (0x0128)

Win3 display start point1 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	win3_dsp_yst1 Win3 vertical start point(y) of the Panel scanning
15:13	RO	0x0	reserved
12:0	RW	0x000a	win3_dsp_xst1 Win3 horizontal start point(x) of the Panel scanning

### VOP\_WIN3\_SRC\_ALPHA\_CTRL

Address: Operational Base + offset (0x012c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	win3_fading_value
23:16	RW	0x00	win3_src_global_alpha layer0 src global alpha (eused by fading value)
15:9	RO	0x0	reserved
8:6	RW	0x0	win3_src_factor_m0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	win3_src_alpha_cal_m0
4:3	RW	0x0	win3_src_blend_m0
2	RW	0x0	win3_src_alpha_m0
1	RW	0x0	win3_src_color_m0
0	RW	0x0	win3_src_alpha_en

**VOP\_WIN3\_MST2**

Address: Operational Base + offset (0x0130)

Win3 memory start address2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win3_mst2 Win3 frame buffer memory start address2 *must be aliased to 8byte address

**VOP\_WIN3\_DSP\_INFO2**

Address: Operational Base + offset (0x0134)

Win3 display width2/height2 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:16	RW	0x0ef	win3_dsp_height2 Win3 display window height2 win3_dsp_height0 = size -1
15:12	RO	0x0	reserved
11:0	RW	0x13f	win3_dsp_width2 Win3 display window width2 win3_dsp_width = size -1

**VOP\_WIN3\_DSP\_ST2**

Address: Operational Base + offset (0x0138)

Win3 display start point2 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	win3_dsp_yst2 Win3 vertical start point(y) of the Panel scanning
15:13	RO	0x0	reserved
12:0	RW	0x000a	win3_dsp_xst2 Win3 horizontal start point(x) of the Panel scanning

**VOP\_WIN3\_DST\_ALPHA\_CTRL**

Address: Operational Base + offset (0x013c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8:6	RW	0x0	win3_dst_factor_m0
5:0	RW	0x00	win3_dst_factor_reserved

**VOP\_WIN3\_MST3**

Address: Operational Base + offset (0x0140)

Win3 memory start address3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	win3_mst3 Win3 frame buffer memory start address3 *must be aliased to 8byte address

**VOP\_WIN3\_DSP\_INFO3**

Address: Operational Base + offset (0x0144)

Win3 display width3/height3 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:16	RW	0x0ef	win3_dsp_height3 Win3 display window height3 win3_dsp_height0 = size -1
15:12	RO	0x0	reserved
11:0	RW	0x13f	win3_dsp_width3 Win3 display window width3 win3_dsp_width = size -1

**VOP\_WIN3\_DSP\_ST3**

Address: Operational Base + offset (0x0148)

Win3 display start point3 on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	win3_dsp_yst3 Win3 vertical start point(y) of the Panel scanning
15:13	RO	0x0	reserved
12:0	RW	0x000a	win3_dsp_xst3 Win3 horizontal start point(x) of the Panel scanning

**VOP\_WIN3\_FADEING\_CTRL**

Address: Operational Base + offset (0x014c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	win3_fading_en
23:16	RW	0x00	win3_fading_offset_b
15:8	RW	0x00	win3_fading_offset_g
7:0	RW	0x00	win3_fading_offset_r

### VOP\_HWC\_CTRL0

Address: Operational Base + offset (0x0150)

Hwc ctrl register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:19	RO	0x0	reserved
18	RW	0x0	hwc_lut_en
17:15	RO	0x0	reserved
14	RW	0x0	hwc_endian_swap hwc 8pp palette data Big-endian/ Little-endian select 1'b0 : Big-endian 1'b1 : Little-endian
13	RW	0x0	hwc_alpha_swap hwc RGB alpha swap 1'b0 : ARGB 1'b1 : RGBA
12	RW	0x0	hwc_rb_swap hwc RGB Red and Blue swap 1'b0 : RGB 1'b1 : BGR
11	RO	0x0	reserved
10	RW	0x0	hwc_csc_mode hwc RGB2YUV Color space conversion: 1'b0 : no CSC 1'b1 : RGB2YUV
9	RW	0x0	hwc_no_outstanding hwc AXI master read outstanding 1'b0 : enable 1'b1 : disable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	RW	0x0	hwc_interlace_read hwc interlace read mode 1'b0 : disable 1'b1 : enable
7	RO	0x0	reserved
6:5	RW	0x0	hwc_size 2'b00 : 32x32 2'b01 : 64x64 2'b10 : 96x96 2'b11 : 128x128
4	RW	0x0	hwc_mode hwc color mode 1'b0 : normal color mode 1'b1 : reversed color mode
3:1	RW	0x0	hwc_data_fmt
0	RW	0x0	hwc_en

**VOP\_HWC\_CTRL1**

Address: Operational Base + offset (0x0154)

Hwc ctrl register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6:4	RW	0x0	win3_axi_gather_num
3:1	RO	0x0	reserved
0	RW	0x0	win3_axi_gather_en

**VOP\_HWC\_MST**

Address: Operational Base + offset (0x0158)

Hwc memory start address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	hwc_mst HWC data memory start address

**VOP\_HWC\_DSP\_ST**

Address: Operational Base + offset (0x015c)

Hwc display start point on panel

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
28:16	RW	0x000a	hwc_dsp_yst HWC vertical start point(y) of the Panel scanning
15:13	RO	0x0	reserved
12:0	RW	0x000a	hwc_dsp_xst HWC horizontal start point(x) of the Panel scanning

**VOP\_HWC\_SRC\_ALPHA\_CTRL**

Address: Operational Base + offset (0x0160)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	hwc_fading_value
23:16	RW	0x00	hwc_src_global_alpha layer0 src global alpha (eused by fading value)
15:9	RO	0x0	reserved
8:6	RW	0x0	hwc_src_factor_m0
5	RW	0x0	hwc_src_alpha_cal_m0
4:3	RW	0x0	hwc_src_blend_m0
2	RW	0x0	hwc_src_alpha_m0
1	RW	0x0	hwc_src_color_m0
0	RW	0x0	hwc_src_alpha_en

**VOP\_HWC\_DST\_ALPHA\_CTRL**

Address: Operational Base + offset (0x0164)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8:6	RW	0x0	hwc_dst_factor_m0
5:0	RW	0x00	hwc_dst_m0_reserved

**VOP\_HWC\_FADING\_CTRL**

Address: Operational Base + offset (0x0168)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	hwc_fading_en
23:16	RW	0x00	hwc_fading_offset_b
15:8	RW	0x00	hwc_fading_offset_g
7:0	RW	0x00	hwc_fading_offset_r

**VOP\_HWC\_RESERVED1**

Address: Operational Base + offset (0x016c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	reserved

**VOP\_POST\_DSP\_HACT\_INFO**

Address: Operational Base + offset (0x0170)

post scaler down horizontal start and end

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	dsp_hact_st_post Panel display scanning horizontal active start point
15:13	RO	0x0	reserved
12:0	RW	0x014a	dsp_hact_end_post Panel display scanning horizontal active end point

**VOP\_POST\_DSP\_VACT\_INFO**

Address: Operational Base + offset (0x0174)

Panel active horizontal scanning start point and end point

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	dsp_vact_st_post Panel display scanning horizontal active start point
15:13	RO	0x0	reserved
12:0	RW	0x00fa	dsp_vact_end_post Panel display scanning horizontal active end point

**VOP\_POST\_SCL\_FACTOR\_YRGB**

Address: Operational Base + offset (0x0178)

post\_yrgb scaling factor

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x1000	post_vs_factor_yrgb post YRGB vertical scaling factor: factor=((src_height[31:16]) /(dst_height[31:16]))*2^12
15:0	RW	0x1000	post_hs_factor_yrgb Post YRGB horizontal scaling factor: factor=((src_width[15:0]) /(dst_width[15:0]))*2^12

### VOP\_POST\_RESERVED

Address: Operational Base + offset (0x017c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00001000	post_reserved

### VOP\_POST\_SCL\_CTRL

Address: Operational Base + offset (0x0180)

post scaling start point offset

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	post_ver_sd_en 1'b0 : post ver scl down disable 1'b1 : post ver scl down enable
0	RW	0x0	post_hor_sd_en 1'b0 : post hor scl down disable 1'b1 : post hor scl down enable

### VOP\_POST\_DSP\_VACT\_INFO\_F1

Address: Operational Base + offset (0x0184)

Panel active horizontal scanning start point and end point F1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	dsp_vact_st_post Panel display scanning horizontal active start point
15:13	RO	0x0	reserved
12:0	RW	0x00fa	dsp_vact_end_post Panel display scanning horizontal active end point

### VOP\_DSP\_HTOTAL\_HS\_END

Address: Operational Base + offset (0x0188)

Panel scanning horizontal width and hsync pulse end point

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x014a	dsp_htotal Panel display scanning horizontal period
15:13	RO	0x0	reserved
12:0	RW	0x000a	dsp_hs_end Panel display scanning hsync pulse width

### VOP\_DSP\_HACT\_ST\_END

Address: Operational Base + offset (0x018c)

Panel active horizontal scanning start point and end point

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	dsp_hact_st Panel display scanning horizontal active start point
15:13	RO	0x0	reserved
12:0	RW	0x014a	dsp_hact_end Panel display scanning horizontal active end point

### VOP\_DSP\_VTOTAL\_VS\_END

Address: Operational Base + offset (0x0190)

Panel scanning vertical height and vsync pulse end point

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x00fa	dsp_vtotal Panel display scanning vertical period.
15:13	RO	0x0	reserved
12:0	RW	0x000a	dsp_vs_end Panel display scanning vsync pulse width

### VOP\_DSP\_VACT\_ST\_END

Address: Operational Base + offset (0x0194)

Panel active vertical scanning start point and end point

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x000a	dsp_vact_st Panel display scanning vertical active start point
15:13	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12:0	RW	0x00fa	dsp_vact_end Panel display scanning vertical active end point

**VOP\_DSP\_VS\_ST\_END\_F1**

Address: Operational Base + offset (0x0198)

Vertical scanning start point and vsync pulse end point of even filed in interlace mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x0000	dsp_vs_st_f1 Panel display scanning vertical vsync start point of 2nd field (interlace display mode)
15:13	RO	0x0	reserved
12:0	RW	0x0000	dsp_vs_end_f1 Panel display scanning vertical vsync end point of 2nd field(interlace display mode)

**VOP\_DSP\_VACT\_ST\_END\_F1**

Address: Operational Base + offset (0x019c)

Vertical scanning active start point and end point of even filed in interlace mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x0000	dsp_vact_st_f1 Panel display scanning vertical active start point of 2nd field (interlace display mode)
15:13	RO	0x0	reserved
12:0	RW	0x0000	dsp_vact_end_f1 Panel display scanning vertical active end point of 2nd field (interlace display mode)

**VOP\_PWM\_CTRL**

Address: Operational Base + offset (0x01a0)

PWM Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	rpt Repeat Counter This field defines the repeated effective periods of output waveform in one-shot mode. The value N means N+1 repeated effective periods.
23:16	RW	0x00	scale Scale Factor This fields defines the scale factor applied to prescaled clock. The value N means the clock is divided by 2*N. If N is 0, it means that the clock is divided by 512(2*256).

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RO	0x0	reserved
14:12	RW	0x2	prescale Prescale Factor This field defines the prescale factor applied to input clock. The value N means that the input clock is divided by $2^N$ .
11:10	RO	0x0	reserved
9	RW	0x0	clk_sel Clock Source Select 0: non-scaled clock is selected as PWM clock source. It means that the prescale clock is directly used as the PWM clock source 1: scaled clock is selected as PWM clock source
8	RW	0x0	lp_en Low Power Mode Enable 0: disabled 1: enabled When PWM channel is inactive state and Low Power Mode is enabled, the path to PWM Clock prescale module is blocked to reduce power consumption.
7:6	RO	0x0	reserved
5	RW	0x0	output_mode PWM Output mode 0: left aligned mode 1: center aligned mode
4	RW	0x0	inactive_pol Inactive State Output Polarity This defines the output waveform polarity when PWM channel is in inactive state. The inactive state means that PWM finishes the complete waveform in one-shot mode or PWM channel is disabled. 0: negative 1: positive
3	RW	0x1	duty_pol Duty Cycle Output Polarity This defines the polarity for duty cycle. PWM starts the output waveform with duty cycle. 0: negative 1: positive

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:1	RW	0x1	pwm_mode PWM Operation Mode 00: One shot mode. PWM produces the waveform within the repeated times defined by PWMx_CTRL_rpt . 01: Continuous mode. PWM produces the waveform continuously 10: Capture mode. PWM measures the cycles of high/low polarity of input waveform. 11: reserved
0	RW	0x0	pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked the one-shot mode, this bit will be cleared at the end of operation

**VOP\_PWM\_PERIOD\_HPR**

Address: Operational Base + offset (0x01a4)

PWM Period Register/High Polarity Capture Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	pwm_period Output Waveform Period/Input Waveform High Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the period of the output waveform. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the bit [31:1] is taken into account and bit [0] always considered as 0.  If PWM is operated at the capture mode, this value indicates the effective high polarity cycles of input waveform. This value is based on the PWM clock.  The value ranges from 0 to (2^32-1).

**VOP\_PWM\_DUTY\_LPR**

Address: Operational Base + offset (0x01a8)

PWM Duty Register/Low Polarity Capture Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	<p>pwm_duty Output Waveform Duty Cycle/Input Waveform Low Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the duty cycle of the output waveform. The PWM starts the output waveform with duty cycle. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the [31:1] is taken into account.</p> <p>If PWM is operated at the capture mode, this value indicates the effective low polarity cycles of input waveform.</p> <p>This value is based on the PWM clock. The value ranges from 0 to (<math>2^{32}-1</math>).</p>

**VOP\_PWM\_CNT**

Address: Operational Base + offset (0x01ac)

PWM Counter Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	<p>pwm_cnt Timer Counter The 32-bit indicates current value of PWM Channel 0 counter. The counter runs at the rate of PWM clock. The value ranges from 0 to (<math>2^{32}-1</math>).</p>

**VOP\_BCSH\_COLOR\_BAR**

Address: Operational Base + offset (0x01b0)

color bar config register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RW	0x000	color_bar_v v color value
21:12	RW	0x000	color_bar_u u color value
11:2	RW	0x000	color_bar_y y color value
1	RO	0x0	reserved
0	RW	0x0	bcsht_en 1'b0 : bcsht bypass 1'b1 : bcsht enable

**VOP\_BCSH\_BCS**

Address: Operational Base + offset (0x01b4)  
 brightness contrast saturation\*contrast config register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RW	0x3	out_mode video out mode config register 2'b00 : black 2'b01 : blue 2'b10 : color bar 2'b11 : normal video
29:20	RW	0x100	sat_con Saturation*Contrast*256 : 0,1.992*1.992
19:17	RO	0x0	reserved
16:8	RW	0x100	contrast Contrast*256 : 0,1.992
7:0	RW	0x00	brightness Brightness : -128,127

**VOP\_BCSH\_H**

Address: Operational Base + offset (0x01b8)  
 sin hue and cos hue config register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:16	RW	0x100	cos_hue cos hue value
15:9	RO	0x0	reserved
8:0	RW	0x000	sin_hue sin hue value

**VOP\_BCSH\_RESERVED**

Address: Operational Base + offset (0x01bc)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	reserved

**VOP\_CABC\_CTRL0**

Address: Operational Base + offset (0x01c0)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	cabc_stage_up

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23:1	RW	0x000000	cabc_calc_pixel_num Field0000 Abstract Field0000 Description
0	RW	0x0	cabc_en 1'b0 : cabc disable 1'b1 : cabc enable

**VOP\_CABC\_CTRL1**

Address: Operational Base + offset (0x01c4)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	cabc_stage_down Field0000 Abstract Field0000 Description
23:1	RW	0x000000	cabc_total_num
0	RO	0x0	reserved

**VOP\_CABC\_GAUSS\_LINE0\_0**

Address: Operational Base + offset (0x01c8)

Register0000 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x15	t_line0_3
23:16	RW	0x11	t_line0_2
15:8	RW	0x09	t_line0_1
7:0	RW	0x03	t_line0_0

**VOP\_CABC\_GAUSS\_LINE0\_1**

Address: Operational Base + offset (0x01cc)

Register0001 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23:16	RW	0x03	t_line0_6
15:8	RW	0x09	t_line0_5
7:0	RW	0x11	t_line0_4

**VOP\_CABC\_GAUSS\_LINE1\_0**

Address: Operational Base + offset (0x01d0)

Register0002 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x1a	t_line1_3
23:16	RW	0x15	t_line1_2
15:8	RW	0x0b	t_line1_1
7:0	RW	0x04	t_line1_0

**VOP\_CABC\_GAUSS\_LINE1\_1**

Address: Operational Base + offset (0x01d4)

Register0003 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23:16	RW	0x04	t_line1_6
15:8	RW	0x0b	t_line1_5
7:0	RW	0x15	t_line1_4

**VOP\_CABC\_GAUSS\_LINE2\_0**

Address: Operational Base + offset (0x01d8)

Register0004 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x15	t_line2_3
23:16	RW	0x11	t_line2_2
15:8	RW	0x09	t_line2_1
7:0	RW	0x03	t_line2_0

**VOP\_CABC\_GAUSS\_LINE2\_1**

Address: Operational Base + offset (0x01dc)

Register0005 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23:16	RW	0x03	t_line2_6

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:8	RW	0x09	t_line2_5
7:0	RW	0x11	t_line2_4

**VOP\_FRC\_LOWER01\_0**

Address: Operational Base + offset (0x01e0)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x1284	lower01_frm1
15:0	RW	0x4821	lower01_frm0

**VOP\_FRC\_LOWER01\_1**

Address: Operational Base + offset (0x01e4)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x2148	lower01_frm3
15:0	RW	0x8412	lower01_frm2

**VOP\_FRC\_LOWER10\_0**

Address: Operational Base + offset (0x01e8)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0xa55a	lower10_frm1
15:0	RW	0x9696	lower10_frm0

**VOP\_FRC\_LOWER10\_1**

Address: Operational Base + offset (0x01ec)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x5aa5	lower10_frm3
15:0	RW	0x6969	lower10_frm2

**VOP\_FRC\_LOWER11\_0**

Address: Operational Base + offset (0x01f0)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0xdeb7	lower11_frm1
15:0	RW	0x7bed	lower11_frm0

**VOP\_FRC\_LOWER11\_1**

Address: Operational Base + offset (0x01f4)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0xed7b	lower11_frm3
15:0	RW	0xb7de	lower11_frm2

**VOP\_FRC\_RESERVED0**

Address: Operational Base + offset (0x01f8)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	Field0000 Abstract Field0000 Description

**VOP\_FRC\_RESERVED1**

Address: Operational Base + offset (0x01fc)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	Field0000 Abstract Field0000 Description

**VOP\_MMU\_DTE\_ADDR**

Address: Operational Base + offset (0x0300)

MMU current page Table address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	MMU_DTE_ADDR Field0000 Abstract Field0000 Description

**VOP\_MMU\_STATUS**

Address: Operational Base + offset (0x0304)

MMU status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
10:6	RO	0x00	PAGE_FAULT_BUS_ID Field0000 Abstract Index of master responsible for last page fault
5	RO	0x0	PAGE_FAULT_IS_WRITE Field0000 Abstract The direction of access for last page fault: 0 = Read 1 = Write
4	RO	0x0	REPLAY_BUFFER_EMPTY Field0000 Abstract The MMU replay buffer is empty
3	RO	0x0	MMU_IDLE Field0000 Abstract The MMU is idle when accesses are being translated and there are no unfinished translated accesses.
2	RO	0x0	STAIL_ACTIVE Field0001 Abstract MMU stall mode currently enabled. The mode is enabled by command
1	RO	0x0	PAGE_FAULT_ACTIVE Field0000 Abstract MMU page fault mode currently enabled . The mode is enabled by command.
0	RO	0x0	PAGING_ENABLED Field0000 Abstract Paging is enabled

**VOP\_MMU\_COMMAND**

Address: Operational Base + offset (0x0308)

MMU command register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2:0	WO	0x0	MMU_CMD Field0000 Abstract MMU_CMD. This can be: 0: MMU_ENABLE_PAGING 1: MMU_DISABLE_PAGING 2: MMU_ENABLE_STALL 3: MMU_DISABLE_STALL 4: MMU_ZAP_CACHE 5: MMU_PAGEFAULT_DONE 6: MMU_FORCE_RESET

**VOP\_MMU\_PAGE\_FAULT\_ADDR**

Address: Operational Base + offset (0x030c)

MMU logical address of last page fault

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	PAGE_FAULT_ADDR Field0000 Abstract address of last page fault

### **VOP\_MMU\_ZAP\_ONE\_LINE**

Address: Operational Base + offset (0x0310)

MMU Zap cache line register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	WO	0x00000000	MMU_ZAP_ONE_LINE Field0000 Abstract address to be invalidated from the page table cache

### **VOP\_MMU\_INT\_RAWSTAT**

Address: Operational Base + offset (0x0314)

MMU raw interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	READ_BUS_ERROR Field0000 Abstract read bus error
0	RW	0x0	PAGEFAULT Field0000 Abstract page fault

### **VOP\_MMU\_INT\_CLEAR**

Address: Operational Base + offset (0x0318)

MMU raw interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	WO	0x0	READ_BUS_ERROR Field0000 Abstract read bus error
0	WO	0x0	PAGEFAULT Field0000 Abstract page fault

### **VOP\_MMU\_INT\_MASK**

Address: Operational Base + offset (0x031c)

MMU raw interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	READ_BUS_ERROR Field0000 Abstract read bus error
0	RW	0x0	PAGE_FAULT Field0000 Abstract page fault

**VOP\_MMU\_INT\_STATUS**

Address: Operational Base + offset (0x0320)

MMU raw interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RO	0x0	READ_BUS_ERROR Field0000 Abstract read bus error
0	RO	0x0	PAGE_FAULT Field0000 Abstract page fault

**VOP\_MMU\_AUTO\_GATING**

Address: Operational Base + offset (0x0324)

mmu auto gating

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	mmu_auto_gating mmu auto gating when it is 1'b1, the mmu will auto gating it self

**VOP\_WIN2\_LUT\_ADDR**

Address: Operational Base + offset (0x0400)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	Field0000 Abstract Field0000 Description

**VOP\_WIN3\_LUT\_ADDR**

Address: Operational Base + offset (0x0800)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	Field0000 Abstract Field0000 Description

**VOP\_HWC\_LUT\_ADDR**

Address: Operational Base + offset (0x0c00)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	Field0000 Abstract Field0000 Description

**VOP\_GAMMA\_LUT\_ADDR**

Address: Operational Base + offset (0x1000)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	Field0000 Abstract Field0000 Description

**VOP MCU BYPASS\_WPORT**

Address: Operational Base + offset (0x2200)

Register0000 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	Field0000 Abstract Field0000 Description

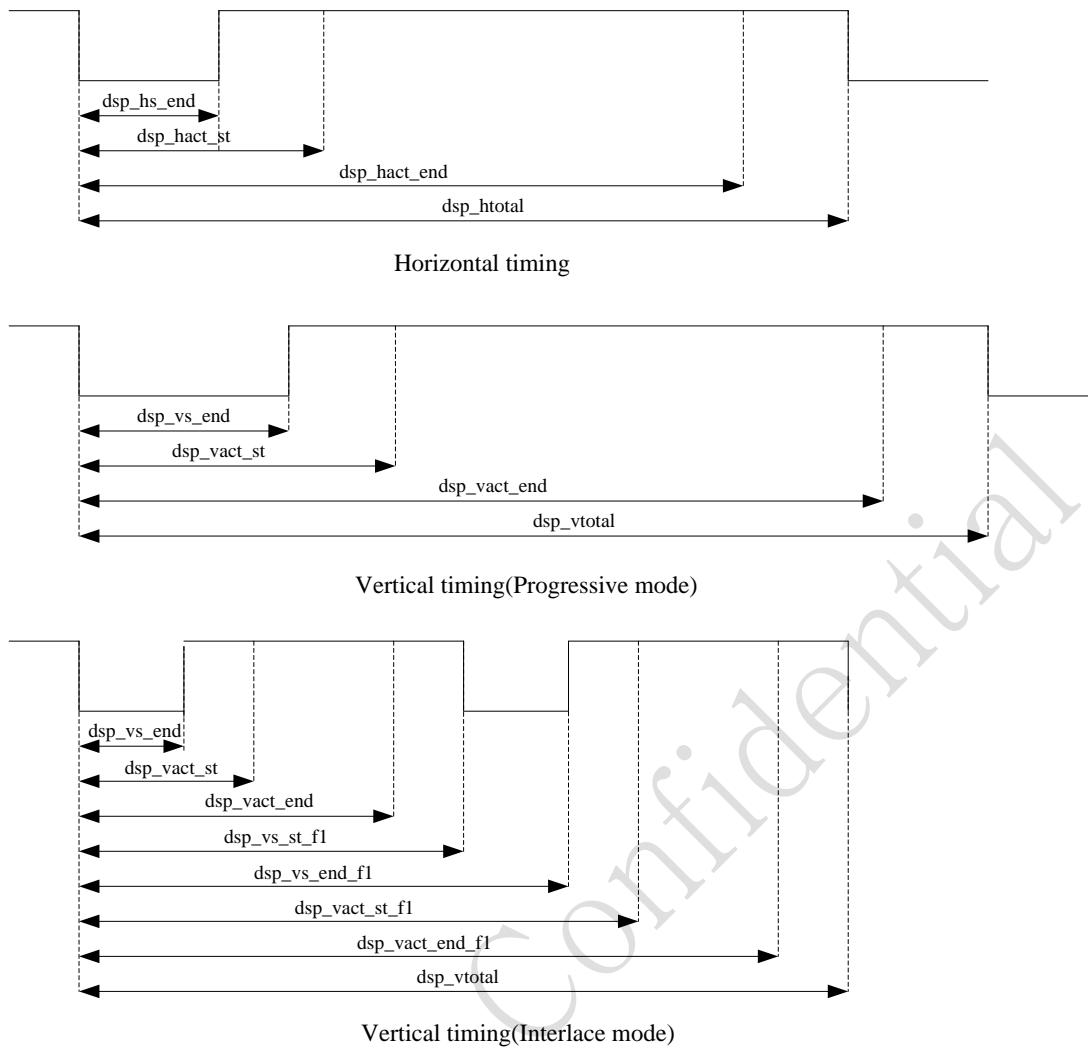
**VOP MCU BYPASS\_RPORT**

Address: Operational Base + offset (0x2300)

Register0001 Abstract

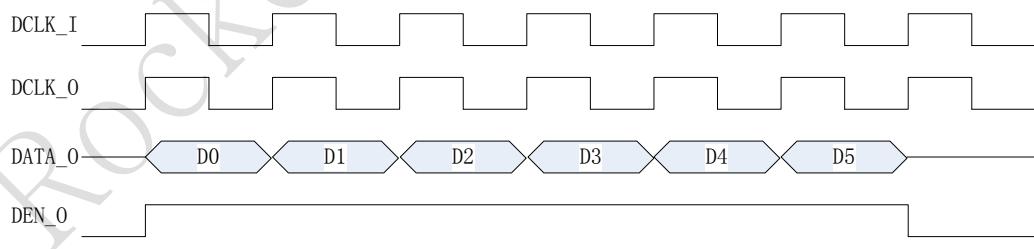
<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	Field0000 Abstract Field0000 Description

**27.5 Timing Diagram****27.5.1 RGB LCD interface timing****1.Timing parameter**

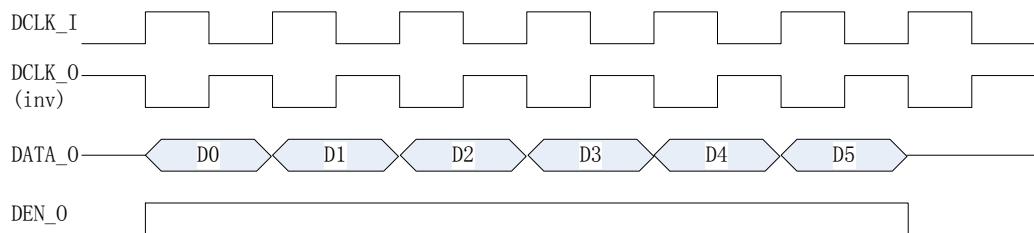


## 2. Data timing for RGB LCD SDR interface

SDR:



SDR+INV:



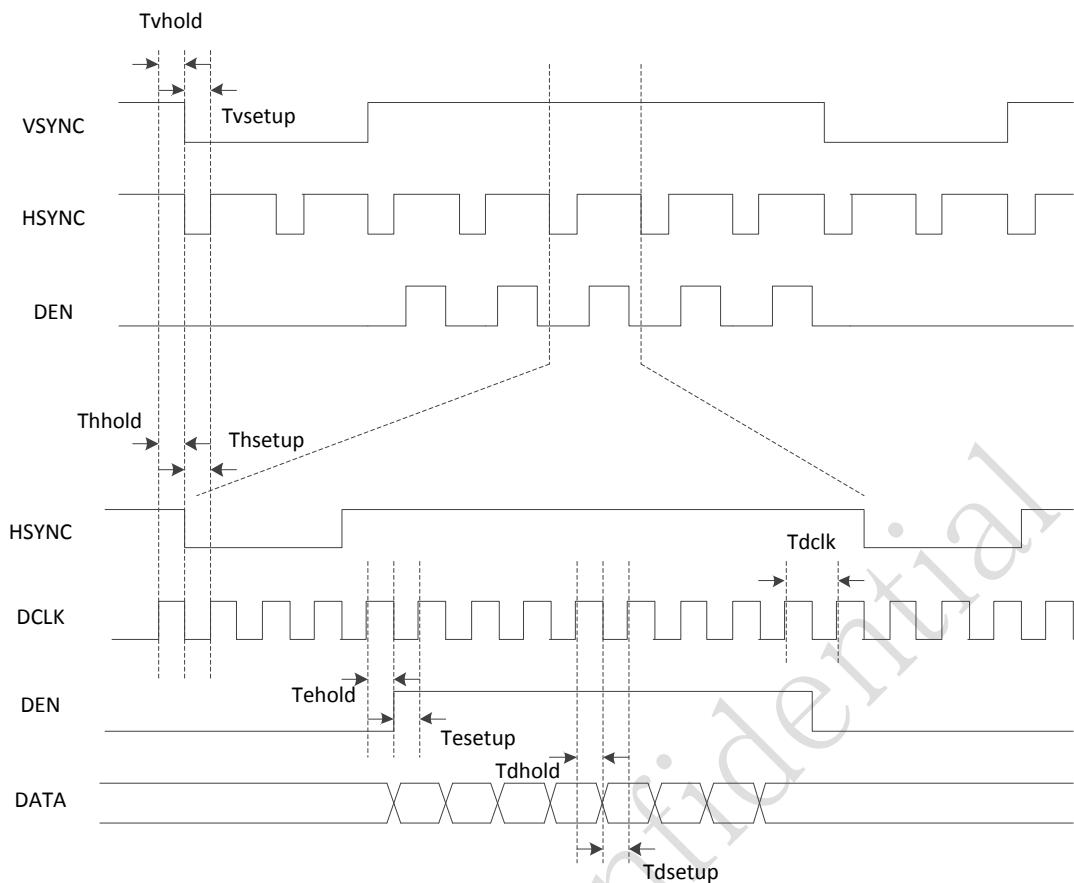


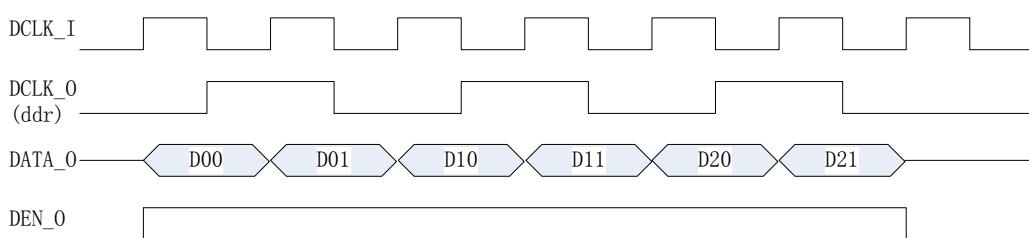
Fig. 27-17 LCDC RGB interface timing (SDR)

Table 27-2 LCDCO RGB interface(SDR) signal timing constant  
( $VDD_{core} = 0.9V$  to  $1.1V$ ,  $VDD_{IO}=3.0V$  to  $3.6V$  ,  $TA = -40^{\circ}C$  ot  $125^{\circ}C$ )

Item	Symbol	Min	Typ	Max	Unit
Display clock period	Tdclk				ns
VSYNC setup to DCLK falling edge	Tvsetup				ns
VSYNC hold from DCLK falling edge	Tvhold				ns
HSYNC setup to DCLK falling edge	Thsetup				ns
HSYNC hold from DCLK falling edge	Thhold				ns
DEN setup to DCLK falling edge	Tesetup				ns
DEN hold from DCLK falling edge	Tehold				ns
DATA setup to DCLK falling edge	Tdsetup				ns
DATA hold from DCLK falling edge	Tdhold				ns

### 3.Data timing for RGB LCD DDR interface

DDR:



## DDR+INV:

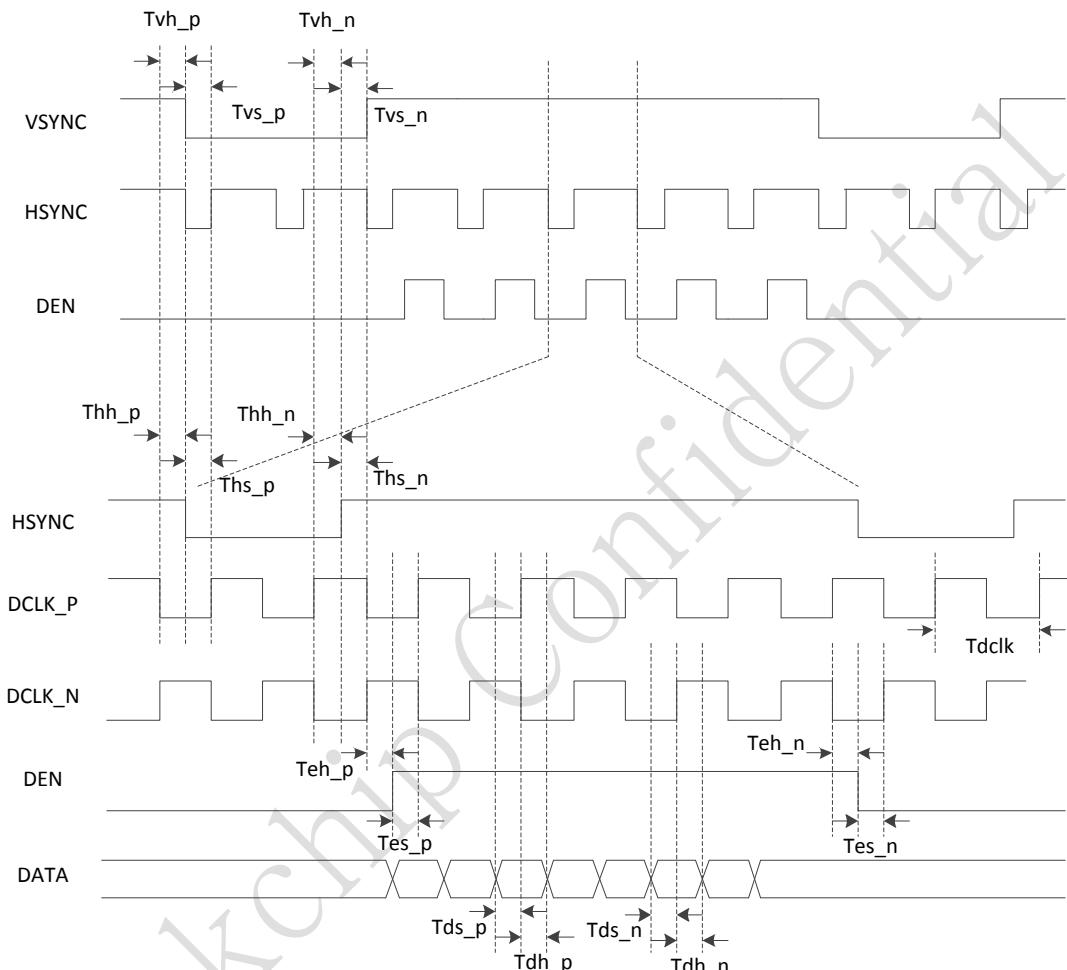
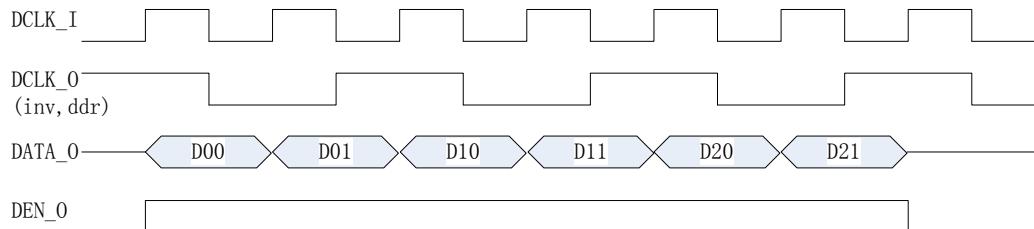


Fig. 27-18 LCDC RGB interface timing (DDR)

Table 27-3 LCDC0 RGB interface (DDR) signal timing constant  
(VDD\_core = 0.9V to 1.1V, VDD\_IO=3.0V to 3.6V , TA = -40°C ot 125°C)

Item	Symbol	Min	Typ	Max	Unit
Display clock period	Tdclk				ns
VSYNC setup to DCLK_N falling edge	Tvs_p				ns
VSYNC hold from DCLK_N falling edge	Tvh_p				ns
HSYNC setup to DCLK_N falling edge	Ths_p				ns
HSYNC hold from DCLK_N falling edge	Thh_p				ns
DEN setup to DCLK_N falling edge	Tes_p				ns
DEN hold from DCLK_N falling edge	The_p				ns
DATA setup to DCLK_N falling edge	Tds_p				ns
DATA hold from DCLK_N falling edge	Tdh_p				ns
VSYNC setup to DCLK_P falling edge	Tvs_p				ns
VSYNC hold from DCLK_P falling edge	Tvh_p				ns
HSYNC setup to DCLK_P falling edge	Ths_p				ns

Hsync hold from DCLK_P falling edge	Thh_p				ns
DEN setup to DCLK_P falling edge	Tes_p				ns
DEN hold from DCLK_P falling edge	The_p				ns
DATA setup to DCLK_P falling edge	Tds_p				ns
DATA hold from DCLK_P falling edge	Tdh_p				ns

Table 27-4 LCDC1 RGB interface signal timing constant  
(VDD\_core = 0.9V to 1.1V, VDD\_IO=3.0V to 3.6V , TA = -40°C ot 125°C)

Item	Symbol	Min	Typ	Max	Unit
Display clock period	Tdclk				ns
VSYNC setup to DCLK falling edge	Tvsetup				ns
VSYNC hold from DCLK falling edge	Tvholt				ns
Hsync setup to DCLK falling edge	Thsetup				ns
Hsync hold from DCLK falling edge	Thhold				ns
DEN setup to DCLK falling edge	Tesetup				ns
DEN hold from DCLK falling edge	Tehold				ns
DATA setup to DCLK falling edge	Tdsetup				ns
DATA hold from DCLK falling edge	Tdhold				ns

## 27.5.2 MCU LCD interface timing

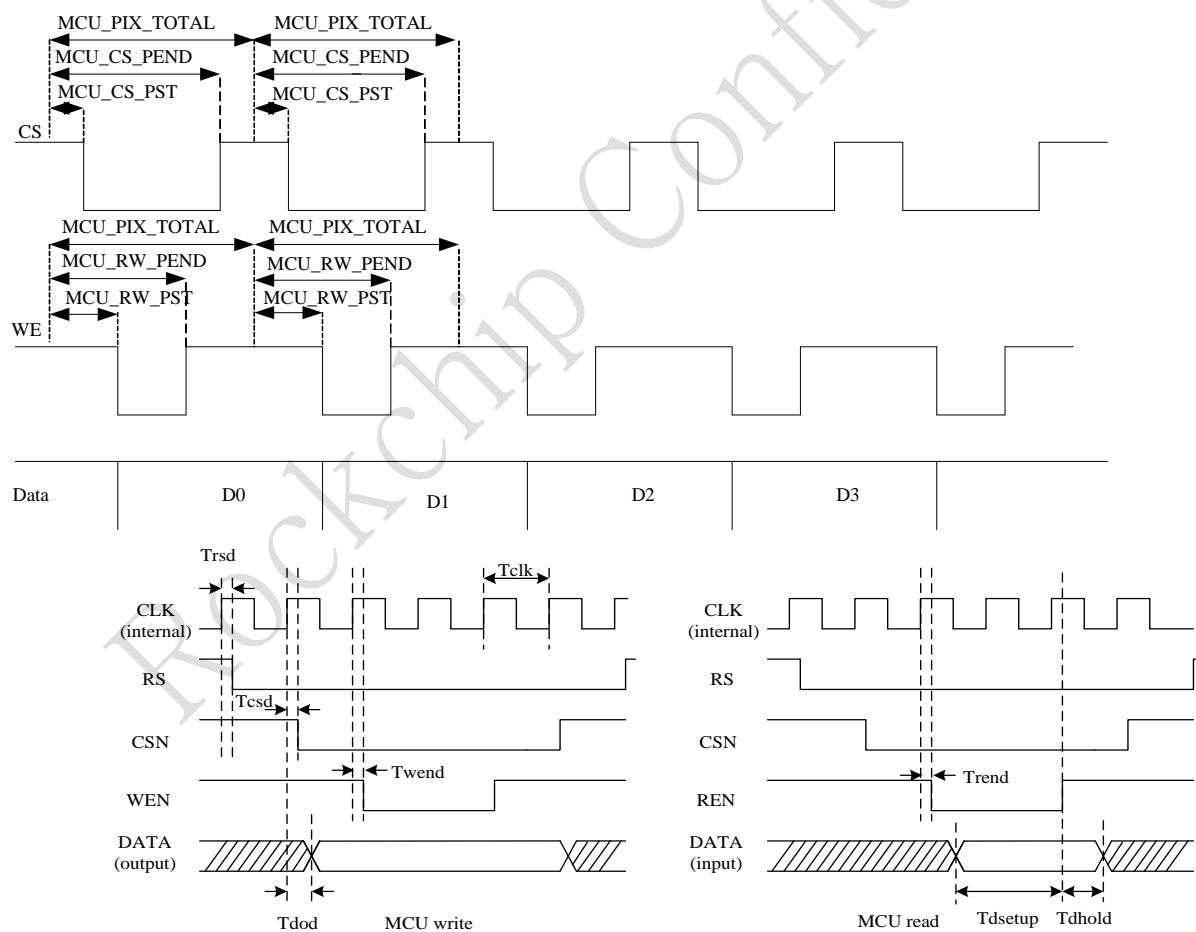


Fig. 27-19 LCDC MCUs (i80) timing

Table 27-5 LCDC0 RGB interface signal timing constant  
(VDD\_core = 0.9V to 1.1V, VDD\_IO=3.0V to 3.6V , TA = -40°C ot 125°C)

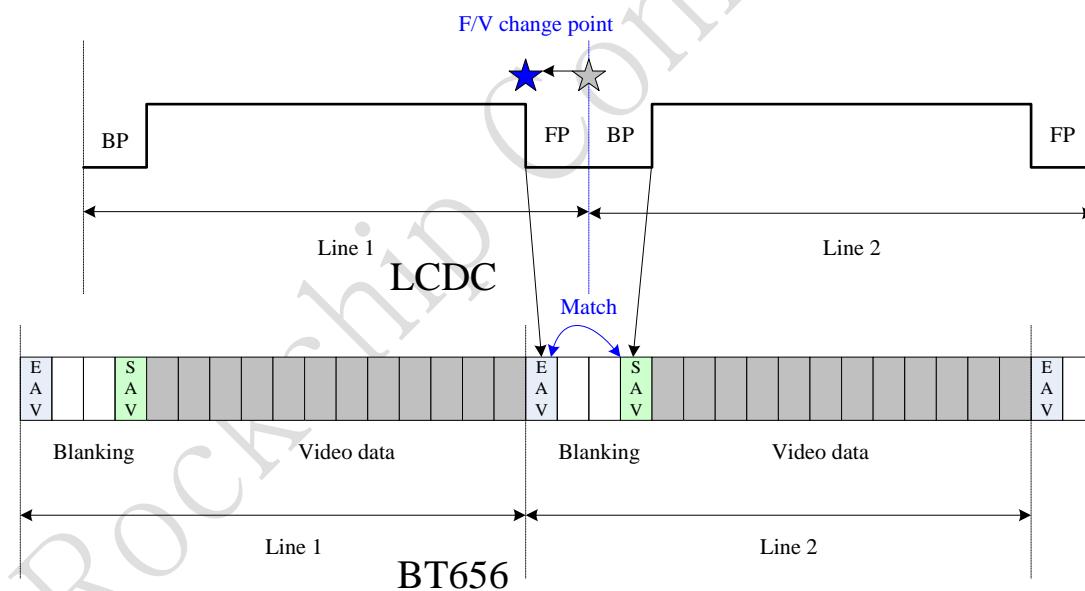
Item	Symbol	Min	Typ	Max	Unit
Internal clock period	Tclk				ns

RS delay from CLK rising edge	Trsd				ns
CSN delay from CLK rising edge	Tcsd				ns
WEN delay from CLK rising edge	Twend				ns
REN delay from CLK rising edge	Trend				ns
D_out delay from CLK rising edge	Tdod				ns
D_in setup to REN rising edge	Tdsetup				ns
D_in hold from REN rising edge	Tdhold				ns

Table 27-6 LCDC1 RGB interface signal timing constant  
(VDD\_core = 0.9V to 1.1V, VDD\_IO=3.0V to 3.6V , TA = -40°C ot 125°C)

Item	Symbol	Min	Typ	Max	Unit
Internal clock period	Tclk				ns
RS delay from CLK rising edge	Trsd				ns
CSN delay from CLK rising edge	Tcsd				ns
WEN delay from CLK rising edge	Twend				ns
REN delay from CLK rising edge	Trend				ns
D_out delay from CLK rising edge	Tdod				ns
D_in setup to REN rising edge	Tdsetup				ns
D_in hold from REN rising edge	Tdhold				ns

### 27.5.3 ITU656 interface timing



## 27.6 Interface Description

### 27.6.1 Display interface description

The VOP is suitable for different display mode by different usage, which is shown as follows.

Display mode	RGB	RGB	RGB	RGB	RGB	RGB
	Parallel 24-bit	Parallel 18-bit	Parallel 16-bit	Serial 2x12-bit	Serial 3x8-bit	Serial 3x8-bit + dummy
DCLK	DCLK	DCLK	DCLK	DCLK	DCLK	DCLK

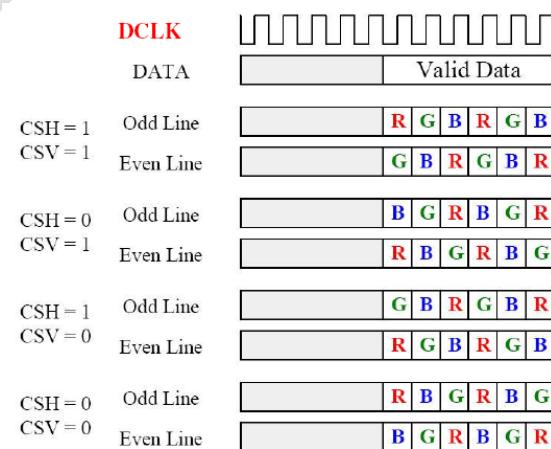
<b>VSYNC</b>	VSYNC	VSYNC	VSYNC	VSYNC	VSYNC	VSYNC
<b>Hsync</b>	Hsync	Hsync	Hsync	Hsync	Hsync	Hsync
<b>DEN</b>	DEN	DEN	DEN	DEN	DEN	DEN
<b>DATA</b>	DATA[23:0]	DATA[17:0]	DATA[15:0]	DATA[11:0]	DATA[7:0]	DATA[7:0]

<b>Display mode</b>	<b>MCU</b>	<b>MCU</b>	<b>MCU</b>	<b>MCU</b>	<b>MCU</b>
	<i>Parallel</i>	<i>Parallel</i>	<i>Parallel</i>	<i>Serial</i>	<i>Serial</i>
	<b>24-bit</b>	<b>18-bit</b>	<b>16-bit</b>	<b>2x12-bit</b>	<b>3x8-bit</b>
<b>DCLK</b>	RS	RS	RS	RS	RS
<b>VSYNC</b>	CSN	CSN	CSN	CSN	CSN
<b>Hsync</b>	WEN	WEN	WEN	WEN	WEN
<b>DEN</b>	REN	REN	REN	REN	REN
<b>DATA</b>	DATA[23:0]	DATA[17:0]	DATA[15:0]	DATA[11:0]	DATA[7:0]

<b>Display mode</b>	<b>ITU656</b>	<b>ITU656</b>	<b>ITU656</b>
	<b>Mode0</b>	<b>Mode1</b>	<b>Mode2</b>
<b>DCLK</b>	DCLK	DCLK	DCLK
<b>VSYNC</b>	-	-	-
<b>Hsync</b>	-	-	-
<b>DEN</b>	-	-	-
<b>DATA</b>	DATA[7:0]	DATA[15:8]	DATA[23:16]

In the case of "RGB serial 3x8-bit", there are four scanning modes for the RGB delta data when delta swap is enable, shown as follows.

<i>RGB delta LCD scanning mode</i>	<i>delta_en</i>	<i>dsp_rg_swap</i>	<i>dsp_rb_swap</i>	<i>dsp_bg_swap</i>
CSH=1,CSV=1	1	0	1	0
CSH=0,CSV=1	1	0	0	0
CSH=1,CSV=0	1	0	0	1
CSH=0,CSV=0	1	0	1	1



## 27.6.2 IOMUX description

There are two VOPs in the chip, config GRF\_SOC\_CON6 register to select a VOP to IO/LVDS port.

programing flow as follow:

step1 : config GRF,iomux,io driver,lc当地 select

GRF\_GPIO1H\_SR and GRF\_GPIO1D\_E register are optional.

config GRF\_BASE + GRF\_SOC\_CON6 = (0x8<<16) | (0x0) to select vop\_big output to IO/LVDS.

config GRF\_BASE + GRF\_SOC\_CON6 = (0x8<<16) | (0x1<<3) to select vop\_lit output to IO/LVDS.

eg:

GRF\_BASE + GRF\_GPIO1D\_IOMUX = ((0x55<<16)|(0x55<<0));

GRF\_BASE + GRF\_GPIO1H\_SR = (0x0f000f00);

GRF\_BASE + GRF\_GPIO1D\_E = (0x00ff00ff);

GRF\_BASE + GRF\_SOC\_CON6 = (0x8<<16) | (0x0);

GRF\_BASE + GRF\_SOC\_CON7 = (0x1fff<<16) | (0x1840);

step 2 : config LVDS PHY0 register to initial LVDS PHY0

eg:

LVDS\_BASE + 0x00\*4 = 0x7f;

LVDS\_BASE + 0x01\*4 = 0x40;

LVDS\_BASE + 0x02\*4 = 0x00;

LVDS\_BASE + 0x03\*4 = 0x46;

LVDS\_BASE + 0x04\*4 = 0x3f;

LVDS\_BASE + 0x05\*4 = 0x3f;

LVDS\_BASE + 0x0d\*4 = 0xa;

step 3 : config LVDS PHY1 register to initial LVDS PHY1

eg:

LVDS\_BASE + 0x40\*4 = 0x7f;

LVDS\_BASE + 0x41\*4 = 0x40;

LVDS\_BASE + 0x42\*4 = 0x00;

LVDS\_BASE + 0x43\*4 = 0x46;

LVDS\_BASE + 0x44\*4 = 0x3f;

LVDS\_BASE + 0x45\*4 = 0x3f;

LVDS\_BASE + 0x4d\*4 = 0xa;

## 27.7 Application Notes

### 27.7.1 DMA transfer mode

There are three DMA transfer modes for loading win0 or win1 frame data determined by following parameters(X=0,1,2,3):

dma\_burst\_length  
winX\_no\_outstanding  
winX\_gather\_en  
winX\_gather\_thres

- **auto outstanding transfer mode(random transfer)**

When winX\_no\_outstanding is 0, multi-bursts transfer command could be sent out to AXI master interface continuously if the internal memory has enough space to store new data. The continuous random burst number is in the range of 1 to 4, mainly depending on the empty level of internal memory, dma\_burst\_length, data format and active image width.

- configured outstanding transfer mode(fixed transfer)**

When `winX_gather_en` is 1, fixed-number of bursts transfer command should be sent out to AXI master interface continuously if the internal memory has enough space to store new data. The fixed-number is determined by `winX_gather_thres`. Since the internal memory size is limited, there is some restriction for the `winX_gather_thres` as follows.

Table 27-7 Gather configuration for all format

Gather Threshold	<code>dma_burst_length =2'b00(burst16)</code>	<code>dma_burst_length =2'b01(burst8)</code>	<code>dma_burst_length =2'b10(burst4)</code>
<b>YUV420</b>	0	0,1,2	0,1,2,3
<b>YUV422</b>			
<b>YUV444</b>			
<b>ARGB888</b>	0,1,2,3	0,1,2,3	0,1,2,3
<b>RGB888</b>			
<b>RGB565</b>			
<b>8BPP</b>	0,1,2,3	0,1,2,3	0,1,2,3
<b>4BPP</b>			
<b>2BPP</b>			
<b>1BPP</b>			

### 27.7.2 Win0/Win1 dma load mode

If you want to improve the efficiency of accessing external memory for loading `winX` frame data, you could assert `winX_dma_load`. When `winX_dma_load` is high, `winX` frame data is loaded in the unit of line composing with one or more burst transfers; otherwise, loaded in the unit of burst transfer. However, it is not suitable for data format YUV420, no-scaling and active width less than 256.

### 27.7.3 IEP direct path

There are two data source for `win0/1/2/3`: external memory and IEP internal memory. However, the IEP data is just active for one layer at one frame time when IEP data path is enable, determined by `dsp_layer0/1/2/3_sel`. Moreover, it is not suitable for `win0/1` with scaling, reverse display.

Direct path interface (DPI) can be used for LCD to display images from other image processing IPs, which also has DPI (slave).

There are programming flows for both DPI display on sequence and DPI display off sequence.

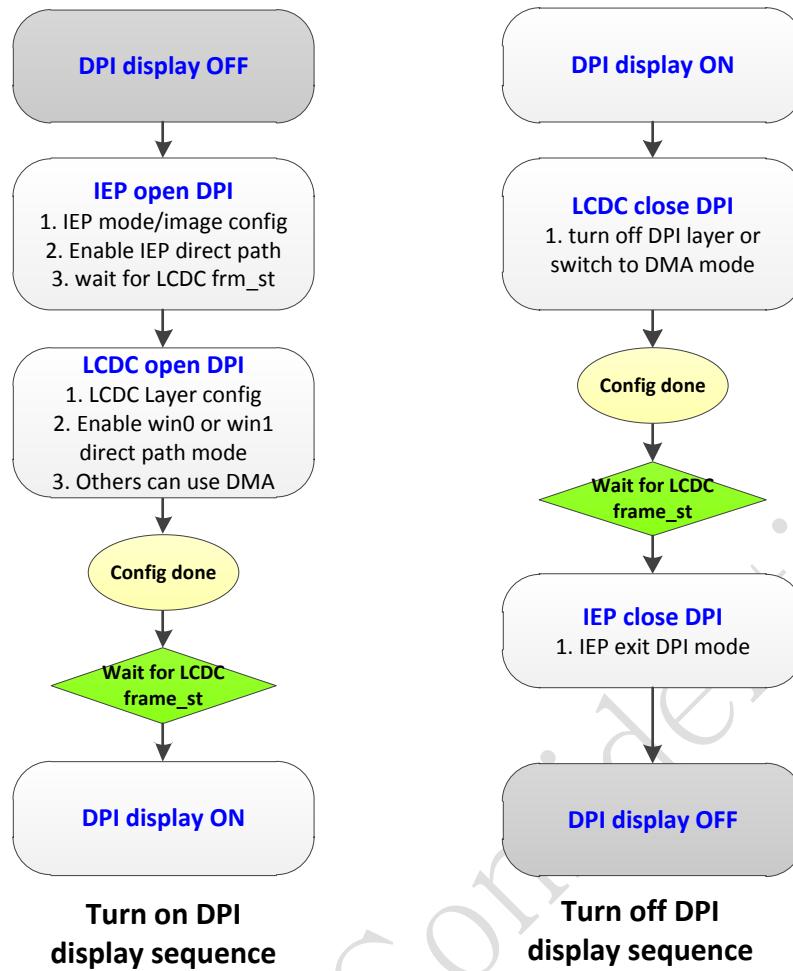


Fig. 27-20 LCDC DPI Programming flow

### 1.Turn on DPI display

First, configure IEP into DPI mode. After doing image information and image processing mode configuration, IEP DPI mode can be turn on for display. IEP is in idle mode only if LCDC's frame start input signal is valid.

Second, configure LCDC for DPI display. Note that only one layer (Win0 or Win1) can use DPI in same frame. Other layers still can use internal DMA.

Finally, set LCDC "config\_done" to confirm all the new configuration and waiting frame sync to start DPI display actually.

### 2.Turn off DPI display

First, close LCDC layer's DPI mode by turning off DPI layer or switching it to DMA mode. Then set LCDC "config\_done" to confirm new configuration.

Second, wait for LCDC's frame synchronization to close DPI display in LCDC.

Finally, turn off IEP's DPI mode.

### 27.7.4 WIN BPP LUT/GAMMA LUT

WIN1 LUT/DSP LUT should be configured before displaying if win2/3\_lut\_en/dsp\_lut\_en is high. You could only update these LUTs by software.

When win1\_lut\_load\_en is 0, the WIN LUT data should be refreshed by software,i.e, writing win1 lut data to the internal memory with the start address WIN1\_LUT\_MST. The memory size is 256x25, i.e, lower 25bits valid, and the writing data number is determined by software, .

When `dsp_lut_load_en` is 0, the DSP LUT data should be refreshed by software,i.e, writing `dsp_lut` data to the internal memory with the start address `DSP_LUT_MST`. The memory size is 256x24, i.e, lower 25bits valid, and the writing data number is determined by software.

### **27.7.5 DMA QoS request**

If you want to get higher priority for VOP to access external memory when the frame data is urgent, a QoS request can be generated and sent out basing on the configured values:

`noc_hurry_en`  
`noc_hurry_value`  
`noc_qos_en`  
`noc_win_qos`

If `noc_qos_en` is enable, a `win0/1_qos_req` is asserted when the empty level of `win0/1`'s linebuffer is greater than the threshold configured in `noc_win_qos`. And it will be disserted when the empty level is smaller than the threshold or `noc_qos_en` is disable.

If `noc_qos_en` is enable, a `win0/1_hurry_req` is asserted when the empty level of `win2/3`'s fifo is greater than the threshold configured in `noc_win_qos`. And it will be disserted when the empty level is smaller than the threshold or `noc_qos_en` is disable.

Either `win0/1_qos_req` or `win2/3_hurry_req` is high, a QoS request will be sent out for VOP.

### **27.7.6 Mirror display**

If Y-Mirror display is enable, the frame data is loaded from last line to first line, where the start address of first pixel in last line is defined in

`WIN0/1_YRGB0_MST/WIN0/1_CBR0_MST/WIN0/1_YRGB1_MST/WIN0/1_CBR_MST/WIN2/3_MST` for `win0/1/2/3` respectively.

Otherwise, the win's frame line data width and virtual stride should be 64bit-aligned for 8bit-RGB/YUV or 128bit-aligned for 10bit YUV if X-Mirror or Y-Mirror display is enable.

### **27.7.7 DDR interface**

LCD DDR interface is just suitable for Parallel RGB LCD panel and Serial RGB LCD 2x12 panel. If LCD DDR interface is enable, the timing parameters for LCD panel should be even.

Otherwise, you can synchronize output clock with VSYNC or HSYNC depending on `dclk_ddr_sync`.

### **27.7.8 Interrupt**

VOP interrupt is comprised of 12 interrupt sources:

frame start interrupt  
line flag interrupt  
bus error interrupt  
win0 empty interrupt  
win1 empty interrupt  
win2 empty interrupt  
win3 empty interrupt  
hwc empty interrupt  
post empty interrupt  
pwm gen interrupt  
irq\_mmu

Every interrupt has independent interrupt enable (`VOP_INT_EN`), interrupt clear (`VOP_INT_CLR`), interrupt status (`VOP_INT_STATUS`).

## 27.7.9 RGB display mode

RGB display mode is used for RGB panel display and CCIR656 output. It is a continuous frames display mode.

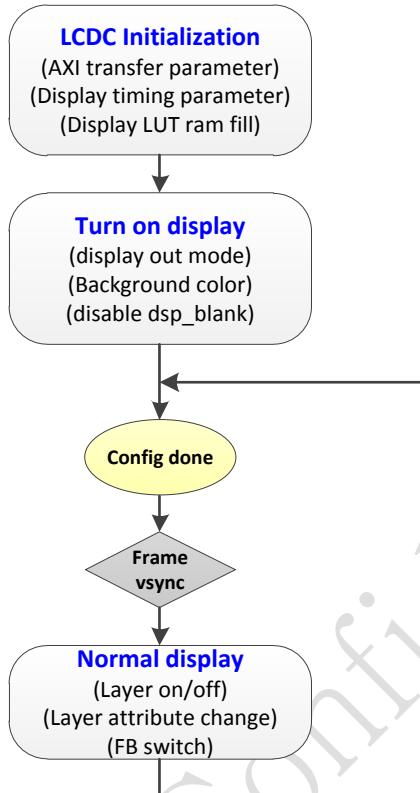


Fig. 27-21 LCDC RGB mode Programming flow

### 1. LCDC initialization

LCDC initialization should be done before turning display on.

First, AXI bus parameter (LCDC\_SYS\_CTRL) should be set for DMA transfer.

Second, display panel/interface timing should be set for display output. The registers are: LCDC\_DSP\_HTOTAL\_HS\_END/ LCDC\_DSP\_HACT\_ST\_END/ LCDC\_DSP\_VTOTAL\_HS\_END/ LCDC\_DSP\_VACT\_ST\_END/ LCDC\_DSP\_VS\_ST\_END\_F1/ LCDC\_DSP\_VACT\_ST\_END\_F1

### 2. Background display

Before normal display, the background display could be turn on.

First, set display output mode (LCDC\_DSP\_CTRL0/1) according to display device.

Second, disable dsp\_blank mode, which would not be enable until frame synchronization.

Finally, writing '1' to "LCDC\_REG\_CFG\_DONE" register then all the frame-sync registers will be enable at the beginning of next frame.

### 3. Normal display

In normal display, all the display layers' attribute could be different according display scenario. So there is a programming loop in this mode.

First, configure all the display layers' attribute registers for the change of image format, location, size, scaling factor, alpha and overlay and so on. Those register would not be enable until frame synchronization.

Finally, write '1' to "LCDC\_REG\_CFG\_DONE" register then all the frame-sync registers will be enable at the beginning of next frame.

### 27.7.10 MCU display mode

MCU display mode is used for MCU panel display or MCU I80 local bus. It is a single frame display mode.

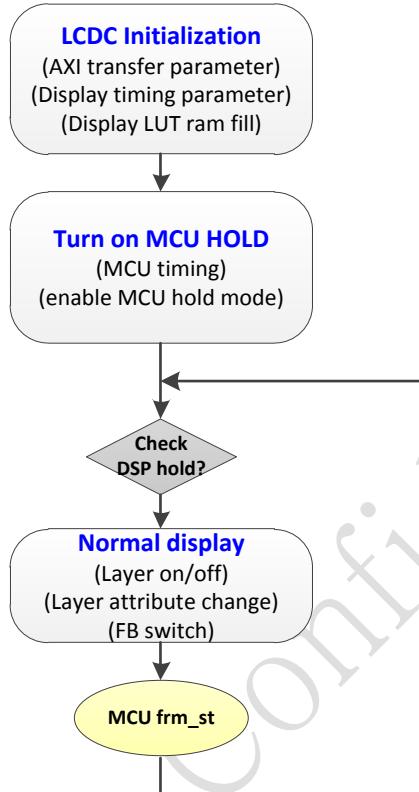


Fig. 27-22 LCDC RGB mode Programming flow

#### 1. LCDC initialization

LCDC initialization should be done before turning display on.

First, AXI bus parameter (LCDC\_SYS\_CTRL) should be set for DMA transfer.

Second, display panel/interface timing should be set for display output. The registers are: LCDC\_DSP\_HTOTAL\_HS\_END/ LCDC\_DSP\_HACT\_ST\_END/ LCDC\_DSP\_VTOTAL\_HS\_END/ LCDC\_DSP\_VACT\_ST\_END/ LCDC\_DSP\_VS\_ST\_END\_F1/ LCDC\_DSP\_VACT\_ST\_END\_F1

Finally, fill the display LUT ram if color LUT function enable.

#### 2. Turn on MCU hold mode

First setting MCU timing parameter (LCDC\_MCU\_CTRL[26:0]).

Turn on MCU hold mode (LCDC\_MCU\_CTRL[31] and LCDC\_MCU\_CTRL[27]), then wait for MCU display hold (read LCDC\_MCU\_CTRL[28] if its value is '1', or set display hold valid interrupt)

#### 3. Single display

If MCU display hold status is valid, single MCU display frame could be start by setting mcu\_frame\_st (LCDC\_MCU\_CTRL[28])

First, configure all the display layers' attribute registers for the change of image format, location, size, scaling factor, alpha and overlay and so on. Those register would not be enable until MCU frame start.

Second, write '1' to start one MCU frame.

Finally, wait for MCU display hold status.

### 27.7.11 MIPI control

#### 1.double channel

MIPI double channel display is supported in the version(only left-right type).

Config doub\_channel\_en register in DSP\_CTRL0 to adapt double channel display .

When doub\_channel mode ,the vop will output two data bus to MIPI PHY,data0 is left panel data,data1 is right panel data.

Double channel overlap display is supported at the same time.

#### normal mode:

left panel data is from 0 to (width/2 -1).

right panel data is from width/2 to width-1.

#### overlap mode:

left panel data is from 0 to (width/2 -1+overlap number).

right panel data is from width/2-overlap number to width-1.

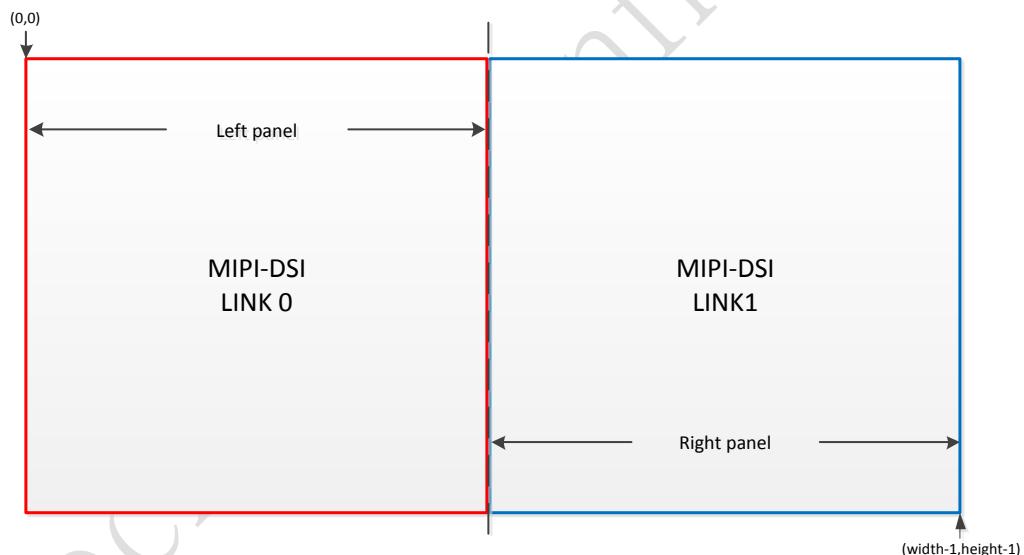


Fig. 27-23 normal mode left-right type display

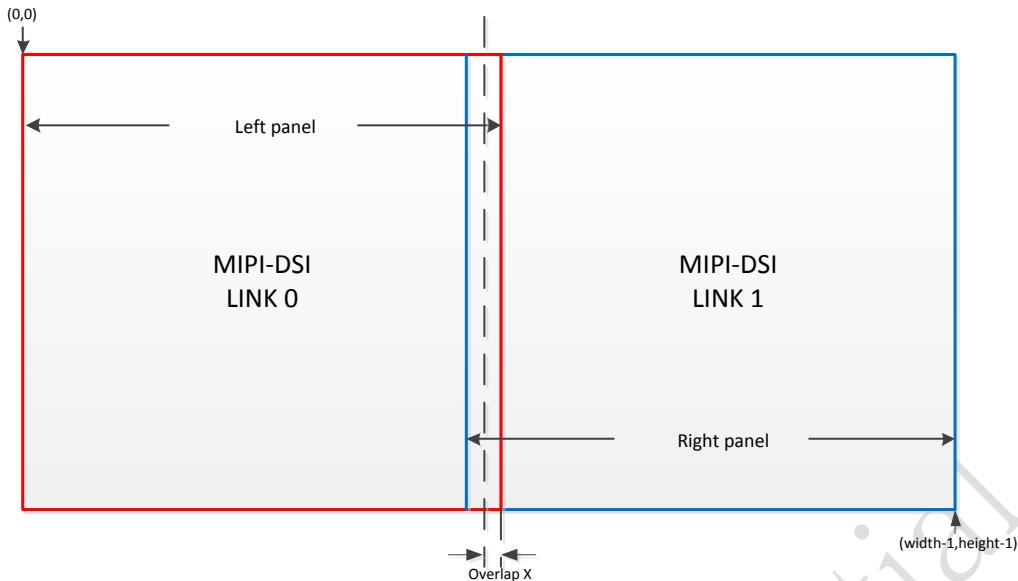


Fig. 27-24 overlap mode left-right type display

The overlap number equal double\_ch\_overlap\_num value \*2,in the range of 0~16.

## 2.halt mode

Mipi halt fuction is supported in this version, detail configuration reference MIPI-DSI chapter.

## 3.command mode flow

Mipi command mode is supported in this version .

There is programming flows for command mode .

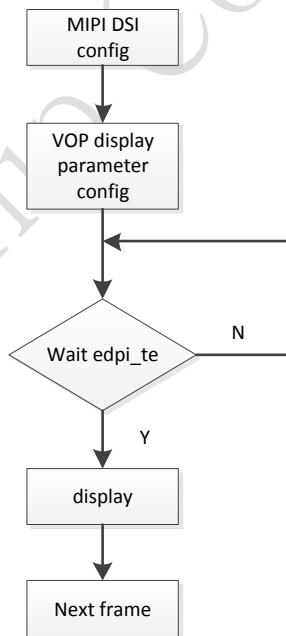


Fig. 27-25 command mode flow

### 27.7.12 Immediately control register

There are two type register in VOP , one type is effective immediately,the other is effective by frame sync.

Effective immediately registers list as follows,other registers are all effective by frame sync.

Table 27-8 effective immediately register table

register address	description
------------------	-------------

0x008[23:21],0x008[15:8]	some dsp ctrl function bit
0x00c	sys ctrl1 register
0x018	background color register
0x01c	mcu ctrl register
0x038	win0 color key register
0x078	win1 color key register
0x0cc	win2 color key register
0x11c	win3 color key register
0x188~0x19c	dsp_timing ctrl registers
0x1a0~0x1a8	pwm ctrl registers
0x1c8~0x1dc	cabc_gauss_parameter registers
0x1ec~0x1f4	frc pattern parameter registers

## Chapter 28 RGA2

### 28.1 Overview

RGA is a separate 2D raster graphic acceleration unit. It accelerates 2D graphics operations, such as point/line drawing, image scaling, rotation, BitBLT, alpha blending and image blur/sharpness.

#### 28.1.1 Features

- **Data format**

- Input data: ARGB/RGB888/RGB565/RGB4444/RGB5551/YUV420/YUV422
- Output data: ARGB/RGB888/RGB565/RGB4444/RGB5551/YUV420/YUV422
- Pixel Format conversion, BT.601/BT.709
- Dither operation
- Max resolution: 8192x8192 source, 4096x4096 destination

- **Scaling**

- Down-scaling: Average filter
- Up-scaling: Bi-cubic filter(source>2048 would use Bi-linear)
- Arbitrary non-integer scaling ratio, from 1/16 to 16

- **Rotation**

- 0, 90, 180, 270 degree rotation
- x-mirror, y-mirror & rotation operation

- **BitBLT**

- Block transfer
- Color palette/Color fill, support with alpha
- Transparency mode (color keying/stencil test, specified value/value range)
- Two source BitBLT:
  - A+B=B only BitBLT, not support scale/rotate mode
  - A+B=C second source (B) has same attribute with (C) plus rotation function

- **Alpha Blending**

- New comprehensive per-pixel alpha(color/alpha channel separately)
- Fading

- **Raster operation**

- ROP2/ROP3/ROP4

- **MMU**

- 4k/64k page size
- Four channel: SRC/SRC1/DST/CMD, individual base address and enable control bit
- TLB pre-fetch

## 28.2 Block Diagram

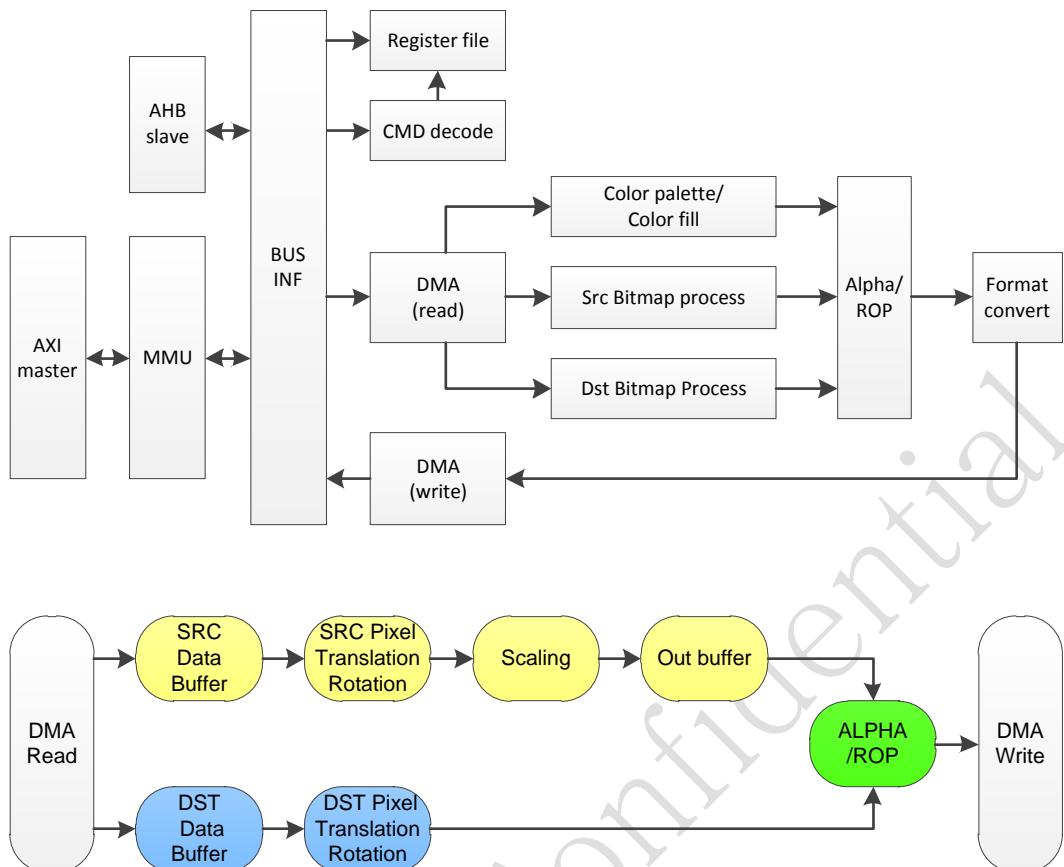


Fig. 28-1 RGA2 Block Diagram

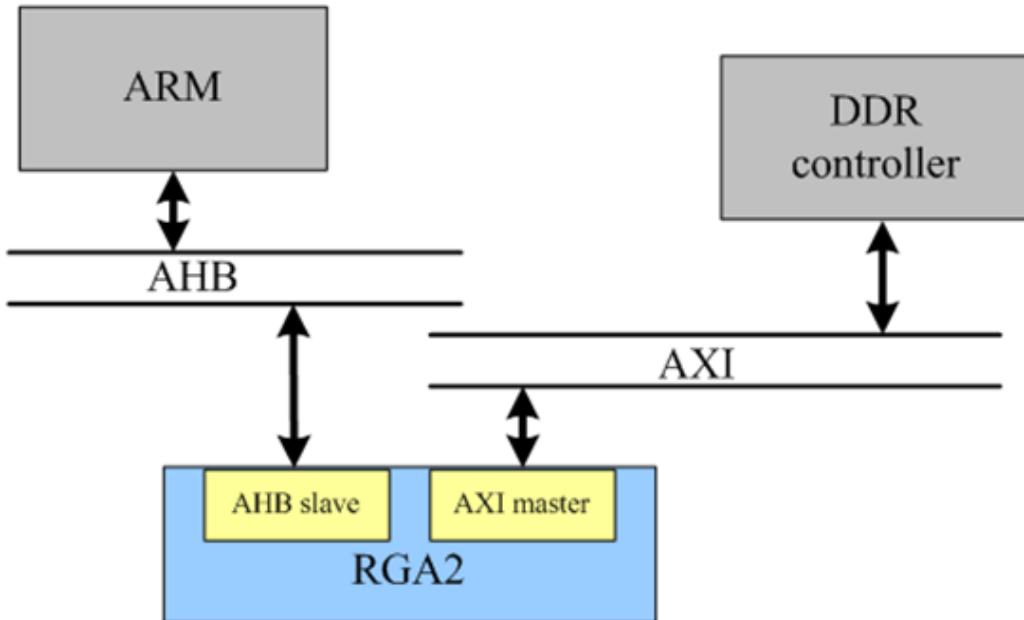


Fig. 28-2 RGA2 in SOC

## 28.3 Function Description

### 28.3.1 Data Format

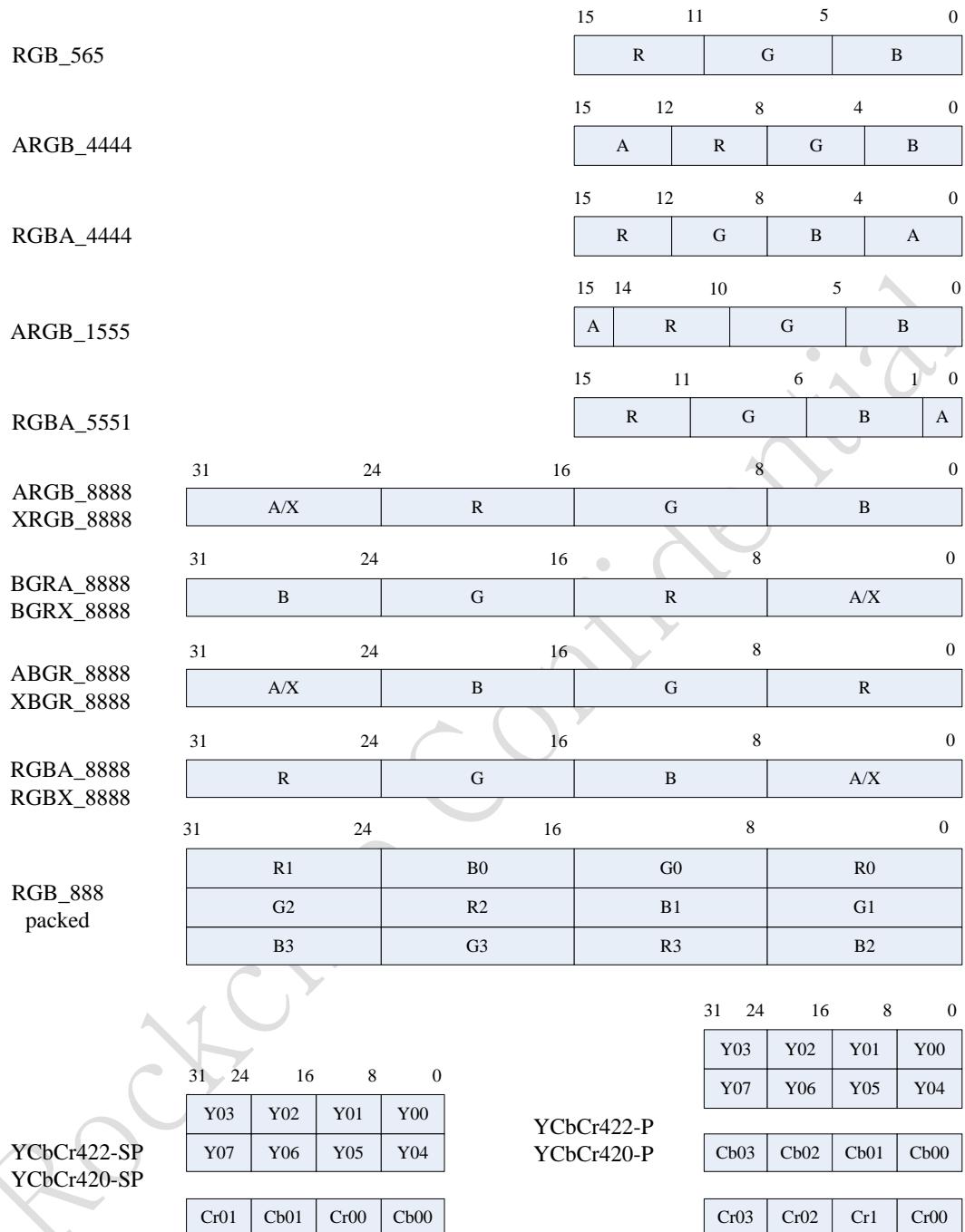


Fig. 28-3 RGA Input Data Format

All input datas (defined by SRC\_IN\_FMT/DST\_IN\_FMT) are converted to ABGR8888. The results are converted to the output data format (defined by DST\_OUT\_FMT).

### 28.3.2 Dithering

There could have dithering operation for source image when the source image format is not RGB565 and the destination format is RGB565.

The down-dithering is done using Dither Allegro.

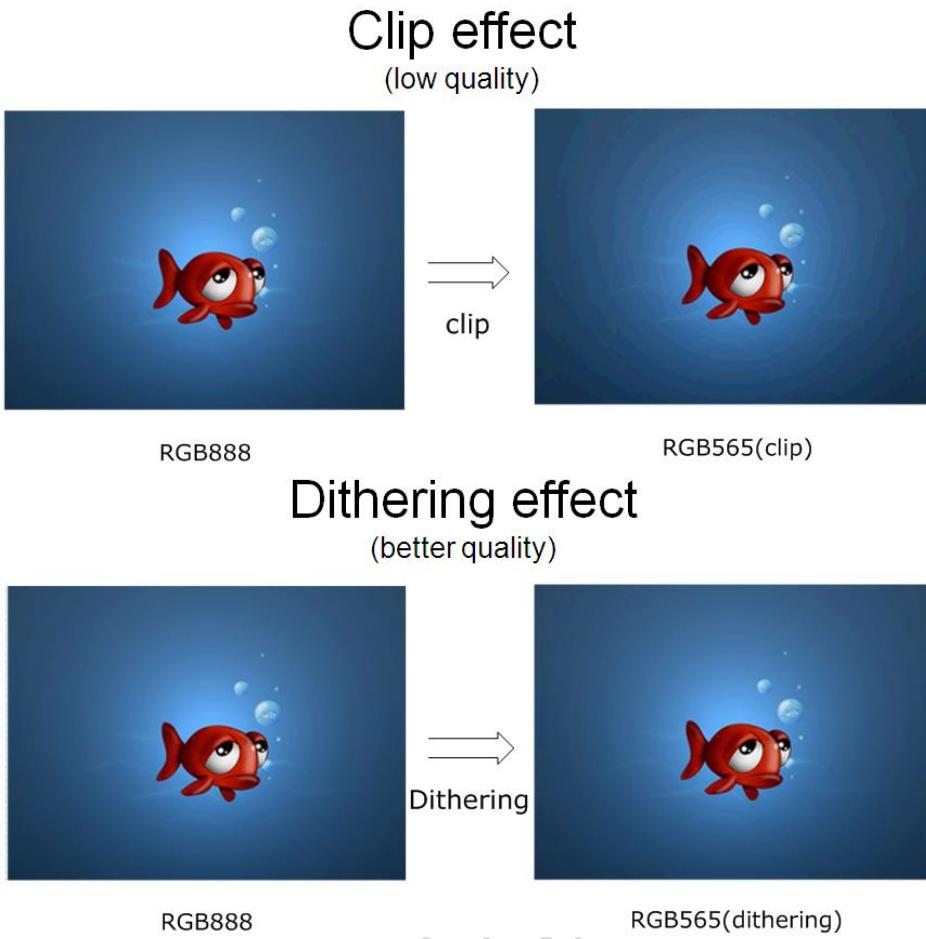
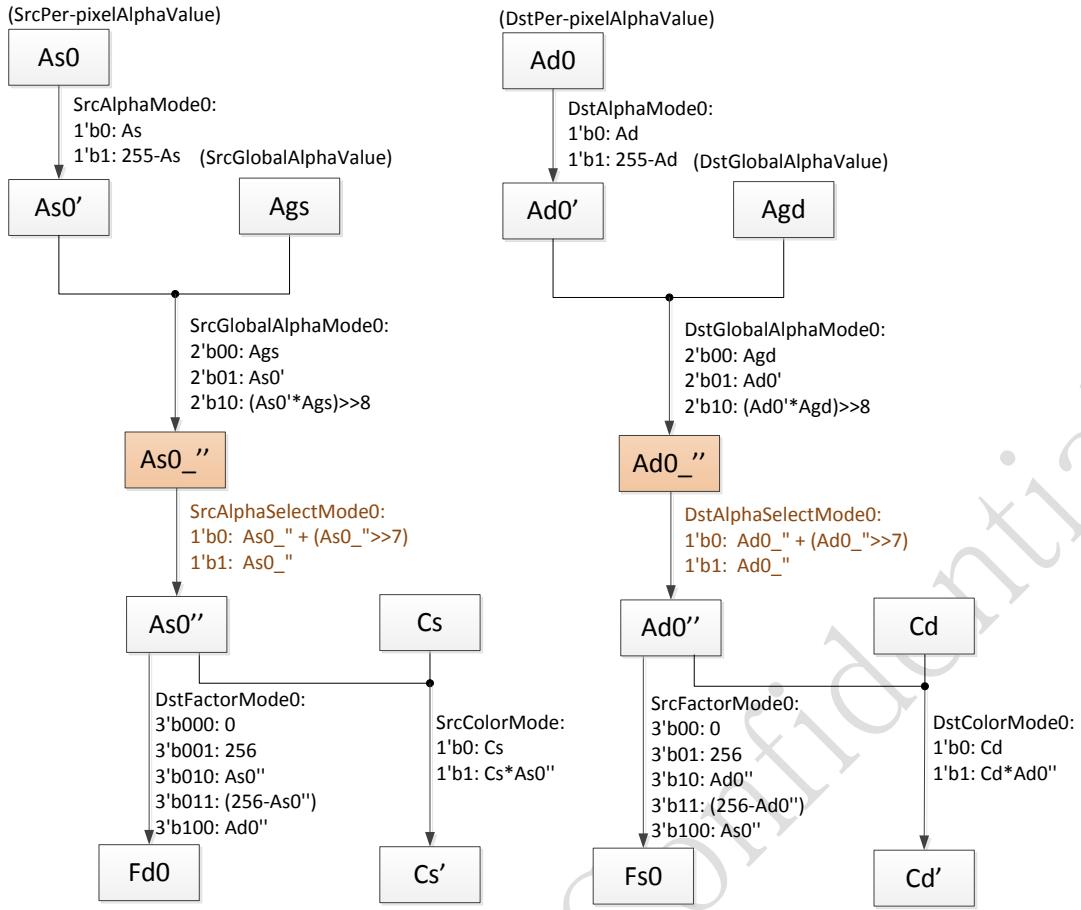


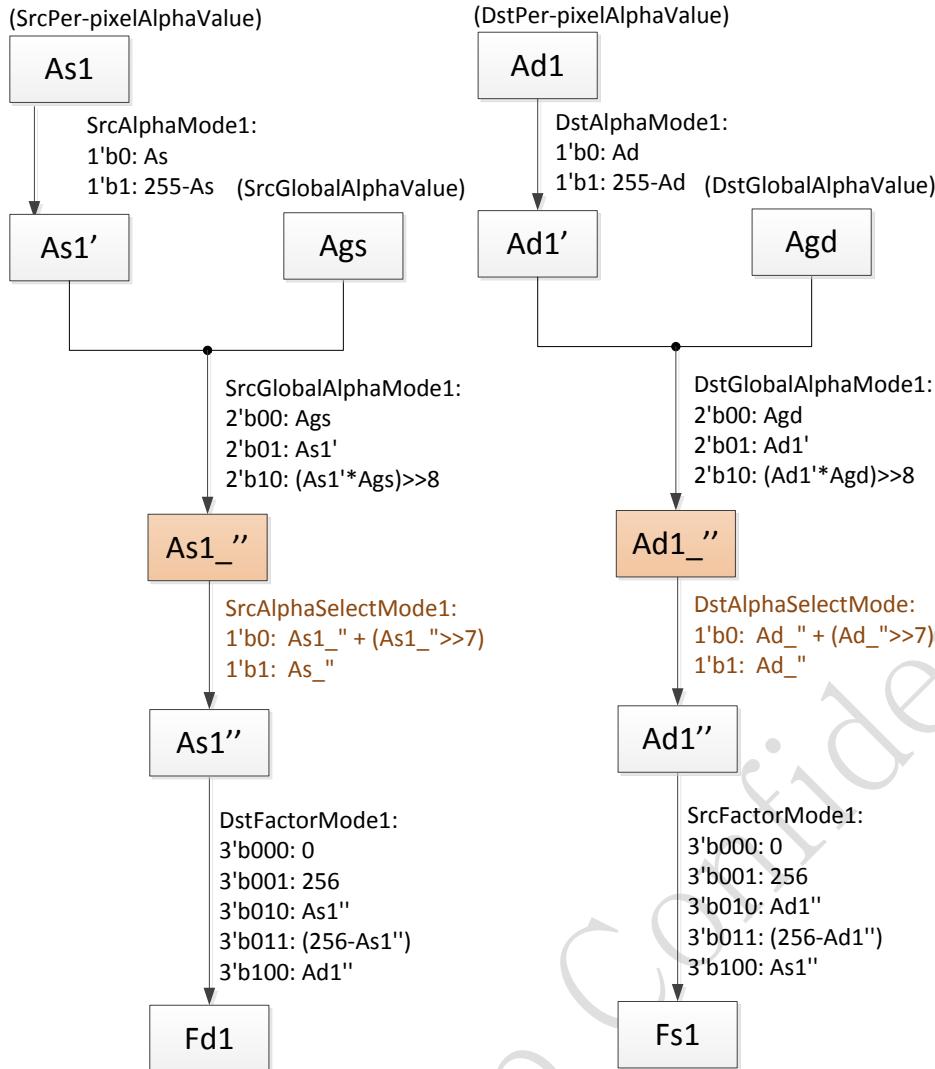
Fig. 28-4 RGA Dither effect

### 28.3.3 Alpha mode



$$Cd = Fs0 * Cs' + Fd0 * Cd' \quad (1)$$

(Cd – dst color, Fs0 – color src factor0, Cs' – src color', Fd0 – color dst factor0, Cd' – dst color')



$$Ad = Fs1 * As1'' + Fd1 * Ad1'' \quad (2)$$

(Ad – dst alpha, Fs1 – alpha src factor1, As1'' – src alpha'', Fd1 – alpha dst factor1, Ad1'' – dst alpha'')

### 28.3.4 Color fill

Two modes of color fill can be done by RGA: solid fill and gradient fill.

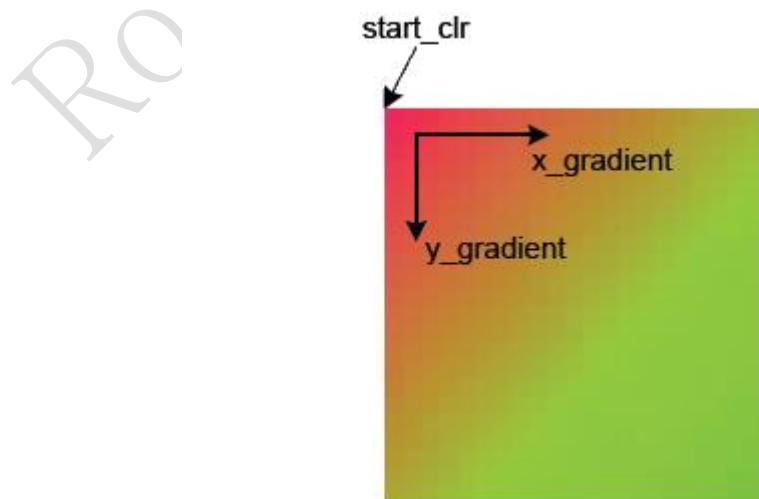


Fig. 28-5 RGA Gradient Fill

Gradient fill using following equations for ARGB calculation of every pixel in different coordinate.

```
A_cur = (A_start + x*x_A_gradient) +y*y_A_gradient;
R_cur = (R_start + x*x_R_gradient) +y*y_R_gradient;
G_cur = (G_start + x*x_G_gradient) +y*y_G_gradient;
B_cur = (B_start + x*x_B_gradient) +y*y_B_gradient;
```

A\_start, R\_start, G\_start, B\_start is the ARGB value of start point. There are four pairs of values for horizontal and vertical gradient. Saturation operation could be enabled or disabled if the color overflows 255 or underflows 0.

### 28.3.5 Raster Operation (ROP)

Raster operation (ROP) is a Boolean operation between operands, which involve AND, OR, XOR, and NOT operations. For ROP2, operands are P (select pan) and D (Destination bitmap). For ROP3, operands are P (pattern), S (source bitmap) and D (Destination bitmap). For ROP4, operands are P (pattern), S (source bitmap), D (Destination bitmap) and MASK.

Table 28-1 RGA ROP Boolean operations

Operator	Meaning
a	Bitwise AND
n	Bitwise NOT (inverse)
o	Bitwise OR
x	Bitwise exclusive OR (XOR)

### 28.3.6 Scaling

The scaling operation is the imageresizing processing of source image. Scaling is done base on ARGB8888 format.

There are three sampling modes: scale down (Average); scale up(Bi-cubic);

## 28.4 Register description

### 28.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
RGA2_RGA_SYS_CTL	0x0000	W	0x00000004	RGA system control register
RGA2_RGA_CMD_CTL	0x0004	W	0x00000000	RGA command control register
RGA2_RGA_CMD_BASE	0x0008	W	0x12345678	RGA command codes base address register
RGA2_RGA_STATUS	0x000c	W	0x00000000	RGA status register
RGA2_RGA_INT	0x0010	W	0x00000000	RGA interrupt register
RGA2_RGA_MMU_CTL	0x0014	W	0x00000000	RGA MMU control 0 register
RGA2_RGA_MMU_CMD_BASE	0x0018	W	0x00000000	Register0000 Abstract

Name	Offset	Size	Reset Value	Description
RGA2_RGA_MODE_CTRL	0x0100	W	0x00000000	RGA mode control register
RGA2_RGA_SRC_INFO	0x0104	W	0x00000000	RGA source information register
RGA2_RGA_SRC_BA_SE0	0x0108	W	0x00000000	source image Y/RGB base address
RGA2_RGA_SRC_BA_SE1	0x010c	W	0x00000000	RGA source image Cb/Cbr base address register
RGA2_RGA_SRC_BA_SE2	0x0110	W	0x00000000	RGA source image Cr base address register
RGA2_RGA_SRC_BA_SE3	0x0114	W	0x00000000	RGA source image 1 base address register
RGA2_RGA_SRC_VIR_INFO	0x0118	W	0x00000000	RGA source image virtual stride / RGA source image tile number
RGA2_RGA_SRC_AC_T_INFO	0x011c	W	0x00000000	RGA source image active width/height register
RGA2_RGA_SRC_X_FACTOR	0x0120	W	0x00000000	RGA source image horizontal scaling factor
RGA2_RGA_SRC_Y_FACTOR	0x0124	W	0x00000000	RGA source image vertical scaling factor
RGA2_RGA_SRC_BG_COLOR	0x0128	W	0x00000000	RGA source image background color
RGA2_RGA_SRC_FG_COLOR	0x012c	W	0x00000000	RGA source image foreground color
RGA2_RGA_CP_GR_A	0x0130	W	0x00000000	RGA source image transparency color min value
RGA2_RGA_SRC_TR_COLOR0	0x0130	W	0x00000000	RGA source image transparency color min value
RGA2_RGA_CP_GR_B	0x0134	W	0x00000000	RGA source image transparency color max value
RGA2_RGA_SRC_TR_COLOR1	0x0134	W	0x00000000	Register0000 Abstract
RGA2_RGA_DST_INFO	0x0138	W	0x00000000	RGA destination format register
RGA2_RGA_DST_BA_SE0	0x013c	W	0x00000000	RGA destination image base address 0 register
RGA2_RGA_DST_BA_SE1	0x0140	W	0x00000000	RGA destination image base address 1 register
RGA2_RGA_DST_BA_SE2	0x0144	W	0x00000000	RGA destination image base address 2 register
RGA2_RGA_DST_VIR_INFO	0x0148	W	0x00000000	RGA destination image virtual width/height register

Name	Offset	Size	Reset Value	Description
RGA2_RGA_DST_AC_T_INFO	0x014c	W	0x00000000	RGA destination image active width/height register
RGA2_RGA_ALPHA_CTRL0	0x0150	W	0x00000000	Alpha control register 0
RGA2_RGA_ALPHA_CTRL1	0x0154	W	0x00000000	Register0000 Abstract
RGA2_RGA_FADING_CTRL	0x0158	W	0x00000000	Fading control register
RGA2_RGA_PAT_CO_N	0x015c	W	0x00000000	Pattern size/offset register
RGA2_RGA_CP_GR_G	0x0160	W	0x00000000	RGA color gradient fill step register (color fill mode)
RGA2_RGA_ROP_CO_N0	0x0160	W	0x00000000	ROP code 0 control register
RGA2_RGA_CP_GR_R	0x0164	W	0x00000000	RGA color gradient fill step register (color fill mode)
RGA2_RGA_ROP_CO_N1	0x0164	W	0x00000000	ROP code 1 control register
RGA2_RGA_MASK_BASE	0x0168	W	0x00000000	RGA mask base address register
RGA2_RGA_MMU_CTL1	0x016c	W	0x00000000	RGA MMU control register 1
RGA2_RGA_MMU_SRC_BASE	0x0170	W	0x00000000	RGA source MMU TLB base address
RGA2_RGA_MMU_SRC1_BASE	0x0174	W	0x00000000	RGA source1 MMU TLB base address
RGA2_RGA_MMU_DST_BASE	0x0178	W	0x00000000	RGA destination MMU TLB base address
RGA2_RGA_MMU_ELS_BASE	0x017c	W	0x00000000	RGA ELSE MMU TLB base address

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

## 28.4.2 Detail Register Description

### RGA2\_RGA\_SYS\_CTRL

Address: Operational Base + offset (0x0000)

RGA system control register

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5	RW	0x0	sw_auto_rst it would auto-resetn after one frame finish. 0: disable 1: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RW	0x0	sw_cclk_sreset_p RGA core clk domain Soft reset, write '1' to this would reset the RGA engine except config registers.
3	WO	0x0	sw_aclk_sreset_p RGA aclk domain Soft reset, write '1' to this would reset the RGA engine except config registers.
2	WO	0x1	sw_auto_ckg RGA auto clock gating enable bit 0: disable 1: enable
1	WO	0x0	sw_cmd_mode RGA command mode 0: slave mode 1: master mode
0	WO	0x0	sw_cmd_op_st_p RGA operation start bit Only used in passive (slave) control mode

**RGA2\_RGA\_CMD\_CTRL**

Address: Operational Base + offset (0x0004)

RGA command control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12:3	RW	0x000	sw_cmd_incr_num RGA command increment number
2	WO	0x0	sw_cmd_stop RGA command stop mode Command execution would stop after the current graphic operation finish if set this bit to 1
1	WO	0x0	sw_cmd_incr_valid_p RGA command increment valid (Auto cleared) When setting this bit, 1. The total cmd number would increase by the RGA_INCR_CMD_NUM. 2. RGA would continue running if idle.
0	RW	0x0	sw_cmd_line_st_p RGA command line fetch start (command line reset) (Auto cleared) When fetch start, the total cmd number would reset to RGA_INCR_CMD_NUM.

**RGA2\_RGA\_CMD\_BASE**

Address: Operational Base + offset (0x0008)

RGA command codes base address register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x12345678	sw_cmd_base RGA command codes base address

**RGA2\_RGA\_STATUS**

Address: Operational Base + offset (0x000c)

RGA status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x000	sw_cmd_total_num RGA command total number
19:8	RO	0x000	sw_cmd_cur_num RGA command current number
7:1	RW	0x00	Reserved Reserved
0	RO	0x0	sw_rga_sta RGA engine status 0: idle 1: working

**RGA2\_RGA\_INT**

Address: Operational Base + offset (0x0010)

RGA interrupt register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved
10	RW	0x0	sw_intr_af_e All command finished interrupt enable
9	RW	0x0	sw_intr_mmu_e MMU interrupt enable
8	RW	0x0	sw_intr_err_e Error interrupt enable
7	WO	0x0	sw_intr_cf_clr Current command finished interrupt clear
6	WO	0x0	sw_intr_af_clr All command finished interrupt clear
5	WO	0x0	sw_intr_mmu_clr MMU interrupt clear
4	WO	0x0	sw_intr_err_clr Error interrupt clear
3	RO	0x0	sw_intr_cf Current command finished interrupt flag
2	RO	0x0	sw_intr_af All command finished interrupt flag

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RO	0x0	sw_intr_mmu MMU interrupt
0	RO	0x0	sw_intr_err Error interrupt flag

**RGA2\_RGA\_MMU\_CTRL0**

Address: Operational Base + offset (0x0014)

RGA MMU control 0 register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RW	0x000000	Reserved
10:9	RW	0x0	sw_els_ch_priority
8:7	RW	0x0	sw_dst_ch_priority
6:5	RW	0x0	sw_src1_ch_priority
4:3	RW	0x0	sw_src_ch_priority
2	RW	0x0	sw_cmd_mmu_flush RGA CMD channel MMU TLB flush: Set 1 to this bit to flush MMU TLB, auto clear
1	RW	0x0	sw_cmd_mmu_en RGA CMD channel MMU enable 0: disable 1: enable
0	RW	0x0	sw_mmu_page_size RGA MMU Page table size 0: 4KB page 1: 64KB page

**RGA2\_RGA\_MMU\_CMD\_BASE**

Address: Operational Base + offset (0x0018)

Register0000 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:0	RW	0x0000000	sw_mmu_cmd_base RGA command MMU TLB base address (word)

**RGA2\_RGA\_MODE\_CTRL**

Address: Operational Base + offset (0x0100)

RGA mode control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RW	0x000000	Reserved
7	RW	0x0	sw_intr_cf_e Current command finished interrupt enable
6	RW	0x0	sw_gradient_sat Gradient saturation calculation mode 0:clip 1:not-clip

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	sw_alpha_zero_key ARGB888 alpha zero key mode 0x000000 would be changed to 0x000100(RGB888)/0x0020(RGB565)for ARGB888 to RGBX/RGB565 color key 0: disable 1: enable
4	RW	0x0	sw_cf_rop4_pat Color fill/ROP4 pattern 0: solid color 1: pattern color
3	RW	0x0	sw_bb_mode Bitblt mode 0: SRC + DST => DST 1: SRC + SRC1 => DST
2:0	RW	0x0	sw_render_mode RGA 2D render mode 000: Bitblt 001: Color palette 010: Rectangle fill 011: Update palette LUT/pattern ram

**RGA2\_RGA\_SRC\_INFO**

Address: Operational Base + offset (0x0104)

RGA source information register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RW	0x00	Reserved
25:24	RW	0x0	sw_bic_coe_sel SRC bicubic scaling coefficient select 00: CATROM 01: MITCHELL 10: HERMITE 11: B-SPLINE
23	RW	0x0	sw_src_dither_up SRC dither up enable 0:disable 1:enable
22:19	RW	0x0	sw_src_trans_e Source transparency enable bits [3]: A value stencil test enable bit [2]: B value stencil test enable bit [1]: G value stencil test enable bit [0]: R value stencil test enable bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
18	RW	0x0	sw_src_trans_mode Source transparency mode 0: normal stencil test (color key) 1: inverted stencil test
17:16	RW	0x0	sw_src_vscl_mode SRC vertical scaling mode 00: no scaling 01: down-scaling 10: up-scaling
15:14	RW	0x0	sw_src_hscl_mode SRC horizontal scaling mode 00: no scaling 01: down-scaling 10: up-scaling
13:12	RW	0x0	sw_src_mir_mode SRC mirror mode 00: no mirror 01: x mirror 10: y mirror 11: x mirror + y mirror
11:10	RW	0x0	sw_src_rot_mode SRC rotation mode 00: 0 degree 01: 90 degree 10: 180 degree 11: 270 degree
9:8	RW	0x0	sw_src_csc_mode Source bitmap YUV2RGB conversion mode 00: BT.601-range0 01: BT.601-range1 10: BT.709-range0 11: BT.709-range1
7	RW	0x0	sw_cp_endian Source Color palette endian swap 0: big endian 1: little endian
6	RW	0x0	sw_src_uvswap Source Cb-Cr swap 0: CrCb 1: CbCr
5	RW	0x0	sw_src_alpha_swap Source bitmap data alpha swap 0: ABGR 1: BGRA

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RW	0x0	sw_src_rbswap Source bitmap data RB swap 0: BGR 1: RGB
3:0	RW	0x0	sw_src_fmt Source bitmap data format 0000: ABGR888 0001: XBGR888 0010: BGR packed 0100: RGB565 0101: ARGB1555 0110: ARGB4444 1000: YUV422SP 1001: YUV422P 1010: YUV420SP 1011: YUV420P 1100: 1BPP (color palette) 1101: 2BPP (color palette) 1110: 4BPP (color palette) 1111: 8BPP (color palette)

**RGA2\_RGA\_SRC\_BASE0**

Address: Operational Base + offset (0x0108)

source image Y/RGB base address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	sw_src_base0 source image Y/RGB base address

**RGA2\_RGA\_SRC\_BASE1**

Address: Operational Base + offset (0x010c)

RGA source image Cb/Cbr base address register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	sw_src_base1 source image Cb base address (YUV422/420-P) source image Cb/Cr base address (YU,V422/420-SP)

**RGA2\_RGA\_SRC\_BASE2**

Address: Operational Base + offset (0x0110)

RGA source image Cr base address register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	sw_src_base2 source image Cr base address (YUV422/420-P)

**RGA2\_RGA\_SRC\_BASE3**

Address: Operational Base + offset (0x0114)

RGA source image 1 base address register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	sw_src_base3 source image 1 RGB base address (source bitblt mode1)

**RGA2\_RGA\_SRC\_VIR\_INFO**

Address: Operational Base + offset (0x0118)

RGA source image virtual stride / RGA source image tile number

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RW	0x00	Reserved
25:16	RW	0x000	sw_mask_vir_stride mask image virtual stride (words)
15	RW	0x0	Reserved
14:0	RW	0x0000	sw_src_act_width source image active width count from 1

**RGA2\_RGA\_SRC\_ACT\_INFO**

Address: Operational Base + offset (0x011c)

RGA source image active width/height register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RW	0x0	Reserved2
28:16	RW	0x0000	sw_src_act_height source image active height
15:13	RW	0x0	Reserved1
12:0	RW	0x0000	sw_src_act_width source image active width

**RGA2\_RGA\_SRC\_X\_FACTOR**

Address: Operational Base + offset (0x0120)

RGA source image horizontal scaling factor

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	sw_src_hsp_factor Source image horizontal up-scaling factor =(DST_ACT_WIDTH/SRC_ACT_WIDTH) * 65536

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	sw_src_hsd_factor Source image horizontal down-scaling factor =(SRC_ACT_WIDTH/DST_ACT_WIDTH) * 65536

**RGA2\_RGA\_SRC\_Y\_FACTOR**

Address: Operational Base + offset (0x0124)

RGA source image vertical scaling factor

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	sw_src_vsp_factor Source image vertical up-scaling factor (DST_ACT_HEIGHT/SRC_ACT_HEIGHT) * 65536
15:0	RW	0x0000	sw_src_vsd_factor Source image vertical down-scaling factor (SRC_ACT_HEIGHT/DST_ACT_HEIGHT) * 65536

**RGA2\_RGA\_SRC\_BG\_COLOR**

Address: Operational Base + offset (0x0128)

RGA source image background color

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	sw_src_bg_color Source image background color ("0" bit color for mono expansion.)

**RGA2\_RGA\_SRC\_FG\_COLOR**

Address: Operational Base + offset (0x012c)

RGA source image foreground color

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	sw_src_fg_color Source image foreground color Source image foreground color ("1" bit color for mono expansion.) Color fill color, Pan color

**RGA2\_RGA\_CP\_GR\_A**

Address: Operational Base + offset (0x0130)

RGA source image transparency color min value

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	sw_gradient_y_a Y gradient value of Alpha (signed 8.8)
15:0	RW	0x0000	sw_gradient_x_a X gradient value of Alpha (signed 8.8)

**RGA2\_RGA\_SRC\_TR\_COLOR0**

Address: Operational Base + offset (0x0130)

RGA source image transparency color min value

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	sw_src_trans_amin source image transparency color A min value
23:16	RW	0x00	sw_src_trans_bmin source image transparency color B min value
15:8	RW	0x00	sw_src_trans_gmin source image transparency color G min value
7:0	RW	0x00	sw_src_trans_rmin source image transparency color R min value

**RGA2\_RGA\_CP\_GR\_B**

Address: Operational Base + offset (0x0134)

RGA source image transparency color max value

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	sw_gradient_y_b Y gradient value of Blue (signed 8.8)
15:0	RW	0x0000	sw_gradient_x_b X gradient value of Blue (signed 8.8)

**RGA2\_RGA\_SRC\_TR\_COLOR1**

Address: Operational Base + offset (0x0134)

Register0000 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	sw_src_trans_amax source image transparency color A max value
23:16	RW	0x00	sw_src_trans_bmax source image transparency color B max value
15:8	RW	0x00	sw_src_trans_gmax source image transparency color G max value
7:0	RW	0x00	sw_src_trans_rmax source image transparency color R max value

**RGA2\_RGA\_DST\_INFO**

Address: Operational Base + offset (0x0138)

RGA destination format register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:19	RW	0x0000	Reserved
18	RW	0x0	sw_dst_csc_clip BGR2YUV Clip mode(from 0~255 clip to 36~235) 1: clip enable; 0: unclip

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
17:16	RW	0x0	sw_dst_csc_mode DST bitmap RGB2YUV conversion mode 00: Bypass 01: BT.601-range0 10: BT.601-range1 11: BT.709-range0
15:14	RW	0x0	sw_dither_mode DST dither down bit mode 00: 888 to 666 01: 888 to 565 10: 888 to 555 11: 888 to 444
13	RW	0x0	sw_dither_down DST dither down enable 0:disable 1:enable
12	RW	0x0	sw_src1_dither_up DST/SRC1 dither up enable 0:disable 1:enable
11	RW	0x0	sw_src1_alpha_swap Source 1 bitmap data alpha swap 0: ABGR 1: BGRA
10	RW	0x0	sw_src1_rbswap Source 1 bitmap data RB swap 0: BGR 1: RGB
9:7	RW	0x0	sw_src1_fmt Source 1 bitmap data format 000: ABGR888 001: XBGR888 010: BGR packed 100: RGB565 101: ARGB1555 110: ARGB4444
6	RW	0x0	sw_dst_uvswap Destination Cb-Cr swap 0: CrCb 1: CbCr
5	RW	0x0	sw_dst_alpha_swap Destination bitmap data alpha swap 0: ABGR 1: BGRA

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RW	0x0	sw_dst_rbswap Destination bitmap data RB swap 0: BGR 1: RGB
3:0	RW	0x0	sw_dst_fmt Destination bitmap data format 0000: ABGR888 0001: XBGR888 0010: BGR packed 0100: RGB565 0101: ARGB1555 0110: ARGB4444 1000: YUV422SP 1001: YUV422P 1010: YUV420SP 1011: YUV420P

**RGA2\_RGA\_DST\_BASE0**

Address: Operational Base + offset (0x013c)

RGA destination image base address 0 register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	sw_dst_base0 destination image Y/RGB base address

**RGA2\_RGA\_DST\_BASE1**

Address: Operational Base + offset (0x0140)

RGA destination image base address 1 register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	sw_dst_base1 destination image Cb/CbCr base address

**RGA2\_RGA\_DST\_BASE2**

Address: Operational Base + offset (0x0144)

RGA destination image base address 2 register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	sw_dst_base2 destination image Cr base address

**RGA2\_RGA\_DST\_VIR\_INFO**

Address: Operational Base + offset (0x0148)

RGA destination image virtual width/height register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RW	0x0	Reserved2
30:16	RW	0x000	sw_src1_vir_stride source image 1 virtual stride (words)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:12	RW	0x0	Reserved1
14:0	RW	0x000	sw_dst_vir_stride destination image virtual stride(words)

**RGA2\_RGA\_DST\_ACT\_INFO**

Address: Operational Base + offset (0x014c)

RGA destination image active width/height register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RW	0x0	Reserved2
27:16	RW	0x000	sw_dst_act_height Destination image active height
15:12	RW	0x0	Reserved1
11:0	RW	0x000	sw_dst_act_width Destination image active width

**RGA2\_RGA\_ALPHA\_CTRL0**

Address: Operational Base + offset (0x0150)

Alpha control register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:21	RW	0x000	Reserved
20	RW	0x0	sw_mask_endian ROP4 mask endian swap 0: big endian 1: little endian
19:12	RW	0x00	sw_dst_global_alpha global alpha value of DST(Agd)
11:4	RW	0x00	sw_src_global_alpha global alpha value of SRC(Ags) fading value in fading mod
3:2	RW	0x0	sw_rop_mode ROP mode select 00: ROP 2 01: ROP 3 10: ROP 4
1	RW	0x0	sw_alpha_rop_sel Alpha or ROP select 0: alpha 1: ROP
0	RW	0x0	sw_alpha_rop_e Alpha or ROP enable 0: disable 1: enable

**RGA2\_RGA\_ALPHA\_CTRL1**

Address: Operational Base + offset (0x0154)

## Register0000 Abstract

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RW	0x0	Reserved Reserved
29	RW	0x0	sw_src_alpha_m1 Src Transparent/opaque of alpha channel (As1') 0: As 1: 255-As
28	RW	0x0	sw_dst_alpha_m1 Dst Transparent/opaque of alpha channel (Ad1') 0: Ad 1: 255-Ad
27:26	RW	0x0	sw_src_blend_m1 Alpha src blend mode select of alpha channel (As1_) 00: Ags 01: As1' 10: (As1'*Ags)>>8 11: reserved
25:24	RW	0x0	sw_dst_blend_m1 Alpha dst blend mode select of alpha channel(Ad1_) 00: Agd 01: Ad1' 10: (Ad1'*Agd)>>8 11: reserved
23	RW	0x0	sw_src_alpha_cal_m1 Alpha src calculate mode of alpha channel(As1") 0: As1"= As1_ "+ (As1_>>7) 1: As1"= As1_ "
22	RW	0x0	sw_dst_alpha_cal_m1 Alpha dst calculate mode of alpha channel(Ad1") 0: Ad1"= Ad1_ "+ (Ad1_>>7) 1: Ad1"= Ad1_ "
21:19	RW	0x0	w_src_factor_m1 Src factore mode of alpha channel(Fs1) 000: 0 001: 256 010: Ad1" 011: 256-Ad1" 100: As1"

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
18:16	RW	0x0	sw_dst_factor_m1 Dst factor mode of alpha channel(Fd1) 000: 0 001: 256 010: As1" 011: 256-As1" 100: Ad1"
15	RW	0x0	sw_src_alpha_m0 Src Transparent/opaque of color channel (As0') 0: As 1: 255-As
14	RW	0x0	sw_dst_alpha_m0 Dst Transparent/opaque of color channel (Ad0') 0: Ad 1: 255-Ad
13:12	RW	0x0	sw_src_blend_m0 Alpha src blend mode select of color channel (As0_) 00: Ags 01: As0' 10: (As0'*Ags)>>8 11: reserved
11:10	RW	0x0	sw_dst_blend_m0 Alpha dst blend mode select of color channel(Ad0_) 00: Agd 01: Ad0' 10: (Ad0'*Agd)>>8 11: reserved
9	RW	0x0	sw_src_alpha_cal_m0 Alpha src calculate mode of color channel(As0") 0: As0"= As0_+ (As0_>>7) 1: As0"= As0_
8	RW	0x0	sw_dst_alpha_cal_m0 Alpha dst calculate mode of color channel(Ad0") 0: Ad0"= Ad0_+ (Ad0_>>7) 1: Ad0"= Ad0_

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:5	RW	0x0	sw_src_factor_m0 Src factor mode of color channel(Fs0) 000: 0 001: 256 010: Ad0'' 011: 256-Ad0'' 100: As0''
4:2	RW	0x0	sw_dst_factor_m0 Dst factor mode of color channel(Fd0) 000: 0 001: 256 010: As0'' 011: 256-As0'' 100: Ad0''
1	RW	0x0	sw_src_color_m0 SRC color select(Cs') 0: Cs 1: Cs * As0''
0	RW	0x0	sw_dst_color_m0 SRC color select(Cd') 0: Cd 1: Cd * Ad0''

**RGA2\_RGA\_FADING\_CTRL**

Address: Operational Base + offset (0x0158)

Fading control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RW	0x00	Reserved
24	RW	0x0	sw_fading_en Fading enable
23:16	RW	0x00	sw_fading_offset_b Fading offset B value
15:8	RW	0x00	sw_fading_offset_g Fading offset G value (Pattern total number when pattern loading)
7:0	RW	0x00	sw_fading_offset_r Fading offset R value (Start point of pattern ram in pattern mode)

**RGA2\_RGA\_PAT\_CON**

Address: Operational Base + offset (0x015c)

Pattern size/offset register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	sw_pat_offset_y Pattern y offset

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23:16	RW	0x00	sw_pat_offset_x Pattern x offset
15:8	RW	0x00	sw_pat_height Pattern height
7:0	RW	0x00	sw_pat_width Pattern width

**RGA2\_RGA\_CP\_GR\_G**

Address: Operational Base + offset (0x0160)

RGA color gradient fill step register (color fill mode)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	sw_gradient_y_g Y gradient value of Green (signed 8.8)
15:0	RW	0x0000	sw_gradient_x_g X gradient value of Green (signed 8.8)

**RGA2\_RGA\_ROP\_CON0**

Address: Operational Base + offset (0x0160)

ROP code 0 control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RW	0x00	Reserved
24:0	RW	0x00000000	sw_rop3_code0 Rop3 code 0 control bits

**RGA2\_RGA\_CP\_GR\_R**

Address: Operational Base + offset (0x0164)

RGA color gradient fill step register (color fill mode)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	sw_gradient_y_r Y gradient value of Red(signed 8.8)
15:0	RW	0x0000	sw_gradient_x_r X gradient value of Red(signed 8.8)

**RGA2\_RGA\_ROP\_CON1**

Address: Operational Base + offset (0x0164)

ROP code 1 control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RW	0x00	Reserved
24:0	RW	0x00000000	sw_rop3_code1 Rop3 code 1 control bits

**RGA2\_RGA\_MASK\_BASE**

Address: Operational Base + offset (0x0168)

RGA mask base address register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	sw_mask_base mask base address in ROP4 mode LUT/ pattern load base address

**RGA2\_RGA\_MMU\_CTRL1**

Address: Operational Base + offset (0x016c)

RGA MMU control register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RW	0x00000	Reserved
13	RW	0x0	sw_els_mmu_flush RGA ELSE channel MMU TLB flush: Set 1 to this bit to flush MMU TLB, auto clear
12	RW	0x0	sw_els_mmu_en RGA ELSE channel MMU enable 0: disable 1: enable
11	RW	0x0	sw_dst_mmu_prefetch_dir 0:forward 1:backward
10	RW	0x0	sw_dst_mmu_prefetch_en 0:disable 1:enable
9	RW	0x0	sw_dst_mmu_flush RGA DST channel MMU TLB flush: Set 1 to this bit to flush MMU TLB, auto clear
8	RW	0x0	sw_dst_mmu_en RGA DST channel MMU enable 0: disable 1: enable
7	RW	0x0	sw_src1_mmu_prefetch_dir 0:forward 1:backward
6	RW	0x0	sw_src1_mmu_prefetch_en 0:disable 1:enable
5	RW	0x0	sw_src1_mmu_flush RGA SRC1 channel MMU TLB flush: Set 1 to this bit to flush MMU TLB, auto clear
4	RW	0x0	sw_src1_mmu_en RGA SRC1 channel MMU enable 0: disable 1: enable
3	RW	0x0	sw_src_mmu_prefetch_dir 0:forward 1:backward

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x0	sw_src_mmu_prefetch_en 0:disable 1:enable
1	RW	0x0	sw_src_mmu_flush RGA SRC channel MMU TLB flush: Set 1 to this bit to flush MMU TLB, auto clear
0	RW	0x0	sw_src_mmu_en RGA SRC channel MMU enable 0: disable 1: enable

**RGA2\_RGA\_MMU\_SRC\_BASE**

Address: Operational Base + offset (0x0170)

RGA source MMU TLB base address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:0	RW	0x0000000	sw_mmu_src_base RGA source MMU TLB base address (128-bit)

**RGA2\_RGA\_MMU\_SRC1\_BASE**

Address: Operational Base + offset (0x0174)

RGA source1 MMU TLB base address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:0	RW	0x0000000	sw_mmu_src1_base RGA source1 MMU TLB base address (128-bit)

**RGA2\_RGA\_MMU\_DST\_BASE**

Address: Operational Base + offset (0x0178)

RGA destination MMU TLB base address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:0	RW	0x0000000	sw_mmu_dst_base RGA destination MMU TLB base address (128-bit)

**RGA2\_RGA\_MMU\_ELS\_BASE**

Address: Operational Base + offset (0x017c)

RGA ELSE MMU TLB base address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:0	RW	0x0000000	sw_mmu_els_base RGA destination MMU TLB base address (128-bit)

## 28.5 Programming Guide

### 28.5.1 Register Partition

There are two types of register in RGA. The first 8 registers (0x0 - 0x1C) are general registers for system configuration including command mode, command parameter, RGA status, general interrupts. The other registers (from 0x100) are command registers for command codes.

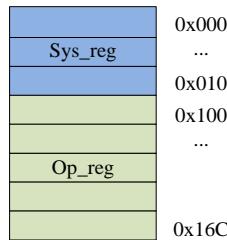


Fig. 28-6 HDMI TX Software Main Sequence Diagram

### 28.5.2 Command Modes

RGA has two command modes: slave mode and master mode. In slave mode (`RGA_SYS_CTRL[1] = 1'b0`), 2D graphic command only could be run one by one. CPU set all the command registers in RGA and then start RGA running by setting `RGA_SYS_CTRL[1]` to '1'. In master mode (`RGA_SYS_CTRL[1] = 1'b1`), 2D graphic commands could be run sequentially. After setting command's number to `RGA_CMD_CTRL[12:3]`, writing '1' to `RGA_CMD_CTRL[0]` will start the command fetch, then Internal command DMA fetch commands from external command line.

Command line is a collection of several command codes with continuous address. At the first start, the command start address (`RGA_CMD_ADDR`) and command number (`RGA_CMD_CTRL[12:3]`) should be set, then write '1' to `cmd_line_st` (`RGA_CMD_CTRL[0]`) to start the command line fetch. Incremental command is supported by setting `cmd_incr_num` (`RGA_CMD_CTRL[12:3]`) and `cmd_incr_valid` (`RGA_CMD_CTRL[1]=1'b1`)

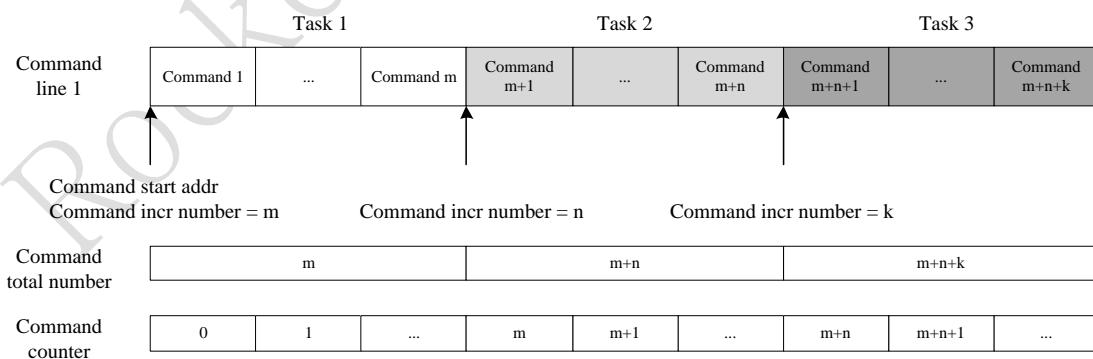


Fig. 28-7 RGA command line and command counter

### 28.5.3 Command Sync

In slave command mode, command sync is controlled by CPU.

In master command mode, user can enable the `current_cmd_int` command by command to

generate a interrupt at the end point of target command operation.

Command 1	Run time	Command 2 (Intr enable)	Run time	Command 3 (Intr disable)	Run time	Command 4 (Intr enable)	Run time	Command 5	Run time
-----------	----------	----------------------------	----------	-----------------------------	----------	----------------------------	----------	-----------	----------

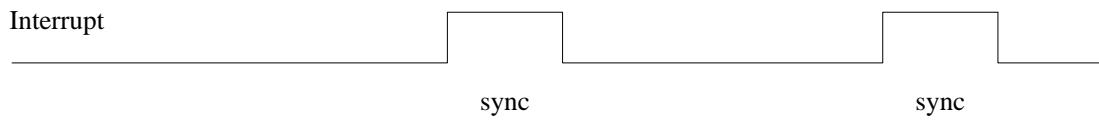


Fig. 28-8 RGA command sync generation

# Chapter 29 Image Enhancement Processor (IEP)

## 29.1 Overview

The Image Enhancement Processor (IEP) receives data from and transmits data to system main memory by AXI bus, or output the data to LCD controller directly.

The features of IEP are as follow:

- **Image format**

- Input data: XRGB/RGB565/YUV420/YUV422
- Output data: ARGB/RGB565/YUV420/YUV422
- ARGB/XRGB/RGB565/YUV swap
- YUV semi-planar/planar
- BT601\_I/BT601\_f/BT709\_I/BT709\_f color space conversion
- RGB dither up/down conversion
- YUV up/down sampling conversion
- Max source image resolution: 8192x8192
- Max scaled image resolution: 4096x4096

- **Enhancement**

- Gamma adjustment with programmable mapping table
- Hue/Saturation/Brightness/Contrast enhancement
- Color enhancement with programmable coefficient
- Detail enhancement with filter matrix up to 9x9
- Edge enhancement with filter matrix up to 9x9
- Programmable difference table for detail enhancement
- Programmable distance table for detail and edge enhancement

- **Noise reduction**

- Compression noise reduction with filter matrix up to 9x9
- Programmable difference table for compression noise reduction
- Programmable distance table for compression noise reduction
- Spatial sampling noise reduction
- Temporal sampling noise reduction
- Optional coefficient for sampling noise reduction

- **High quality scaling**

- Horizontal down-scaling with vertical down-scaling
- Horizontal down-scaling with vertical up-scaling
- Horizontal up-scaling with vertical down-scaling
- Horizontal up-scaling with vertical up-scaling
- Arbitrary non-integer scaling ratio, from 1/16 to 16

- **De-interlace**

- Input 4 fields, output 2 frames mode
- Input 4 fields, output 1 frames mode
- Input 2 fields, output 1 frames mode
- Programmable motion detection coefficient
- Programmable high frequency factor
- Programmable edge interpolation parameter

- Source width up to 1920
- **Interface**
  - Programmable direct path to VOP
  - 32bit AHB bus slave
  - 64bit AXI bus master
  - Combined interrupt output

## 29.2 Block Diagram

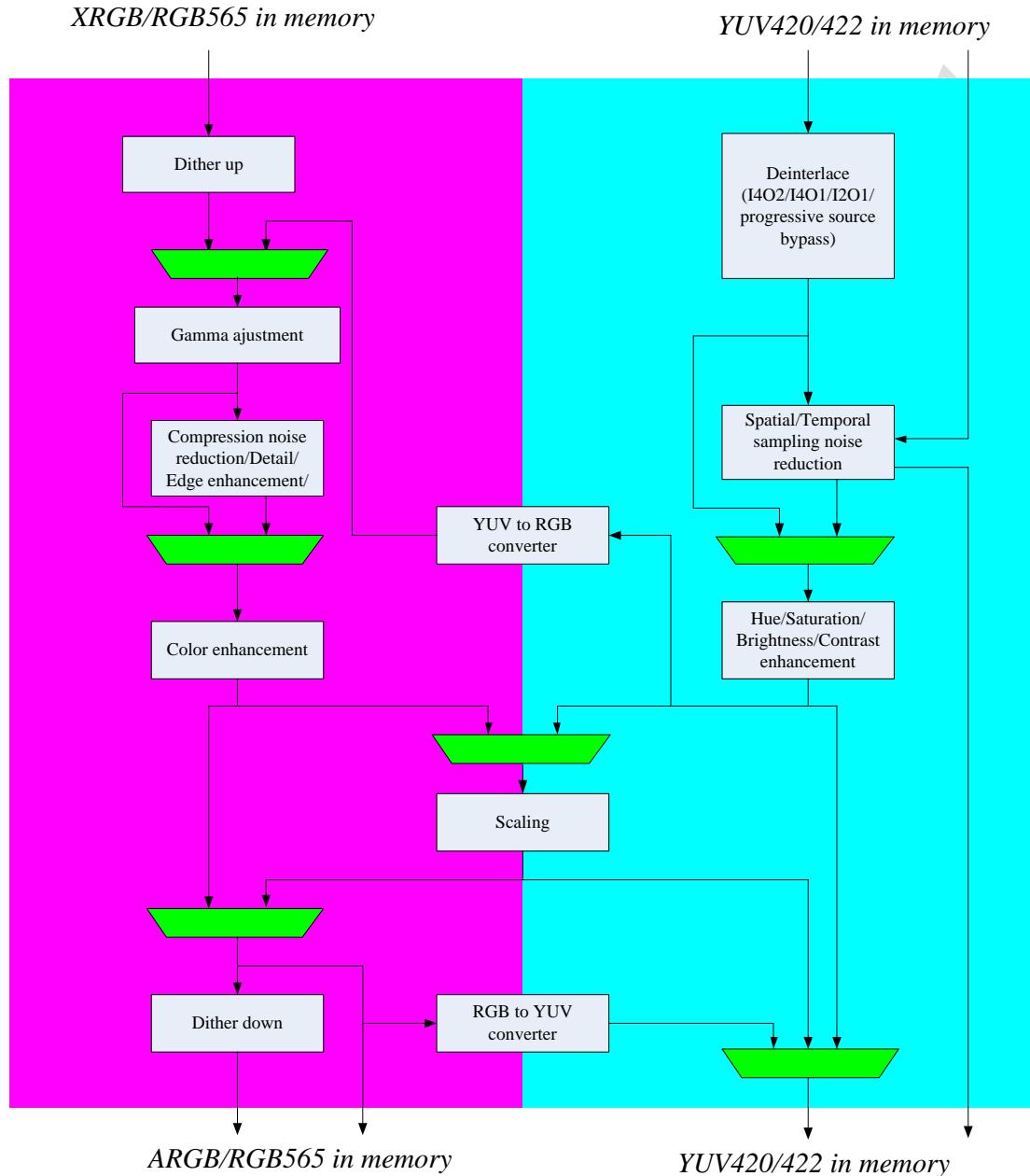


Fig. 29-1 IEP block diagram

The data path in IEP is in the previously diagram. The IEP comprises with:

- Deinterlace

There are five deinterlace mode including I4O2 (input 4 fields and output 2 frames once), I4O1B, I4O1T, I2O1B, I2O1T in the deinterlace block. YUV bypass is also supported.

- Enhancement

Not only hue, saturation, brightness, contrast enhancement, but also blue screen, black screen and color bar are supported in YUV domain enhancement block. Gamma adjustment, edge enhancement, detail enhancement and color enhancement are supported in RGB domain enhancement block.

- Noise Reduction

Spatial and temporal sampling noise can be reduced in YUV domain noise reduction block.

Compression noise can be reduced in RGB domain noise reduction block.

- Scaling

There are four types of scaling modes.

- Horizontal down-scaling with vertical down-scaling
- Horizontal down-scaling with vertical up-scaling
- Horizontal up-scaling with vertical down-scaling
- Horizontal up-scaling with vertical up-scaling

## 29.3 Function description

### 29.3.1 Deinterlace

There are five deinterlace mode including I4O2, I4O1B, I4O1T, I2O1B and I2O1T in the deinterlace block. The I4O2 mode represents for 4 fields of input images and 2 frames of output images, so all of the two groups of source address registers and two groups of destination address registers need to be configured. For example, if source and destination format are both YUV420, the source address register IEP\_SRC\_ADDR\_YRGB, IEP\_SRC\_ADDR\_CBCR are used for source field0 and field 1, the source address register IEP\_SRC\_ADDR\_Y1, IEP\_SRC\_ADDR\_CBCR1 are used for source field2 and field3. The I4O1B and I4O1T mode have the same input images as the I4O2 mode, but only one frame output is generated once. The I2O1B and I2O1T mode have the same output as I4O1B and I4O1T mode, but only two fields input are needed. If bypass mode is selected, there are not any deinterlace operations. The parameter dil\_ei\_sel, dil\_ei\_radius, dil\_ei\_smooth, dil\_ei\_mode, dil\_hf\_en and dil\_hf\_fct in register IEP\_CONFIG0 and registers IEP\_DIL\_MTN\_TAB0~7 may have different influence in deinterlace effect depend on the type of the image source.

### 29.3.2 Noise reduction

Both of spatial and temporal sampling noise reduction are enabled when the 3D denoise bit is set. This function is used for reducing the noise generated at video or picture capturing in the camera sensor. There are four groups of optional noise reduction effect coefficients for luminance and chrominance in spatial and temporal segment.

Compression noise reduction is used for reducing the noise after the decompression of picture or video. Before the compression noise reduction is enabled, the IEP\_ENH\_DDE\_COE0/1 from address 0x400 to 0x5FC for difference and distance coefficients must be written firstly. The filter matrix can be selected from 3x3/5x5/7x7/9x9 and the filter weight can be programmed by configuring IEP\_ENH\_RGB\_CNFG.

### 29.3.3 Enhancement

Not only hue, saturation, brightness, contrast enhancement, but also blue screen, black screen and color bar are supported in this block. IEP\_ENH\_YUV\_CNFG\_0/1/2 registers can be configured to modify the YUV enhance parameters to satisfied with the requirement.

Before the gamma adjustment or contrast enhancement is enabled in RGB domain, the IEP\_ENH\_CG\_TAB from address 0x100 to 0x3FC for B, G, R mapping must be written firstly. If the color enhancement is enabled, the IEP\_ENH\_C\_COE must be written the required value.

Before the edge or detail enhancement is enabled, the IEP\_ENH\_DDE\_COE0/1 from address

0x400 to 0x5FC for difference and distance coefficients must be written firstly. The filter matrix can be selected from 3x3/5x5/7x7/9x9 and the filter weight can be programmed by configuring IEP\_ENH\_RGB\_CNFG.

### 29.3.4 Scaling

There are four types of scaling modes: horizontal down & vertical down-scaling, horizontal down & vertical up-scaling, horizontal up & vertical down-scaling, horizontal up & vertical up-scaling. The down-scaling and up-scaling factor can be got from different way. The detail calculated method is the following:

- vrt\_up\_scl\_fct=floor(src\_image\_height/dst\_image\_height)
- vrt\_dn\_scl\_fct=ceiling((dst\_image\_height+1)/(src\_image\_height+1))
- hrz\_up\_scl\_fct=floor(src\_image\_width/dst\_image\_width)
- hrz\_dn\_scl\_fct=ceiling((dst\_image\_width+1)/(src\_image\_width+1))

There are four up-scaling type (Hermite, Spline, Catrom and Mitchell) can be selected for difference requirement.

### 29.3.5 Format conversion

The color space conversion either from RGB to YUV or from YUV to RGB has the selections including BT601/709\_L/F mode, and the input can be clipped or not.

If the source format is RGB565, dither up must be enabled. In contrary to the destination format is RGB565, dither down must be enabled.

### 29.3.6 Shadow registers

The configuration registers can be configured at any time, but they cannot have any effect immediately unless config\_done is available and a new frame\_start is enabled. The registers IEP\_RAW\_CONFIG0/1, IEP\_RAW\_VIR\_IMG\_WIDTH, IEP\_RAW\_IMG\_SCL\_FCT, IEP\_RAW\_SRC\_IMG\_SIZE, IEP\_RAW\_ENH\_YUV\_CNFG\_0/1/2 corresponding to the registers have the similar names but without letters \_RAW. They are used for raw register value reading before the configurations really have effect on the new frame.

### 29.3.7 VOP direct path

The IEP\_DST\_ADDR for DMA writing is useless if vop\_path\_en bit is set, because all RGB or YUV data is supplied for VOP directly from local bus via VOP and IEP.

## 29.4 Register description

### 29.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
IEP_CONFIG0	0x00000	W	0x00000000	configuration register0
IEP_CONFIG1	0x00004	W	0x00000000	configuration register1
IEP_STATUS	0x00008	W	0x00000000	status register
IEP_INT	0x0000c	W	0x00000000	interrupt register
IEP_FRM_START	0x00010	W	0x00000000	frame start
IEP_CONFIG_DONE	0x00018	W	0x00000000	configuration done
IEP_FRM_CNT	0x0001c	W	0x00000000	frame counter

Name	Offset	Size	Reset Value	Description
IEP_VIR_IMG_WIDT_H	0x00020	W	0x01400140	Image virtual width
IEP_IMG_SCL_FCT	0x00024	W	0x20002000	scaling factor
IEP_SRC_IMG_SIZE	0x00028	W	0x00f00140	Source image width/height
IEP_DST_IMG_SIZE	0x0002c	W	0x00f00140	Destination image width/height
IEP_DST_IMG_WIDT_H_TILE0	0x00030	W	0x00000000	Destination image tile0 width
IEP_DST_IMG_WIDT_H_TILE1	0x00034	W	0x00000000	Destination image tile1 width
IEP_DST_IMG_WIDT_H_TILE2	0x00038	W	0x00000000	Destination image tile2 width
IEP_DST_IMG_WIDT_H_TILE3	0x0003c	W	0x00000000	Destination image tile3 width
IEP_ENH_YUV_CNFG_0	0x00040	W	0x00000000	brightness,contrast,saturation adjustment
IEP_ENH_YUV_CNFG_1	0x00044	W	0x00000000	Hue configuration
IEP_ENH_YUV_CNFG_2	0x00048	W	0x00000000	color bar configuration
IEP_ENH_RGB_CNFG	0x0004c	W	0x00000000	enhancement RGB configuration
IEP_ENH_C_COE	0x00050	W	0x00000000	rgb color enhancement coefficient
IEP_RAW_CONFIG0	0x00058	W	0x00000000	configuration register0
IEP_RAW_CONFIG1	0x0005c	W	0x00000000	configuration register1
IEP_RAW_VIR_IMG_WIDTH	0x00060	W	0x01400140	Image virtual width
IEP_RAW_IMG_SCL_FCT	0x00064	W	0x20002000	scaling factor
IEP_RAW_SRC_IMG_SIZE	0x00068	W	0x00f00140	Source image width/height
IEP_RAW_DST_IMG_SIZE	0x0006c	W	0x00f00140	Destination image width/height
IEP_RAW_ENH_YUV_CNFG_0	0x00070	W	0x00000000	brightness,contrast,saturation adjustment
IEP_RAW_ENH_YUV_CNFG_1	0x00074	W	0x00000000	Hue configuration
IEP_RAW_ENH_YUV_CNFG_2	0x00078	W	0x00000000	color bar configuration
IEP_RAW_ENH_RGB_CNFG	0x0007c	W	0x00000000	enhancement RGB configuration

Name	Offset	Size	Reset Value	Description
IEP_SRC_ADDR_YRG_B	0x00080	W	0x00000000	Start address of source image(Y/RGB)
IEP_SRC_ADDR_CBC_R	0x00084	W	0x00000000	Start address of source image(Cb/Cr)
IEP_SRC_ADDR_CR	0x00088	W	0x00000000	Start address of source image(Cr)
IEP_SRC_ADDR_Y1	0x0008c	W	0x00000000	Start address of source image(Y)
IEP_SRC_ADDR_CBC_R1	0x00090	W	0x00000000	Start address of source image(Cb/Cr)
IEP_SRC_ADDR_CR1	0x00094	W	0x00000000	Start address of source image(Cr)
IEP_SRC_ADDR_Y_I_TEMP	0x00098	W	0x00000000	Start address of source image(Y integer part)
IEP_SRC_ADDR_CBC_R_ITEMP	0x0009c	W	0x00000000	Start address of source image(CBCR integer part)
IEP_SRC_ADDR_CR_ITEMP	0x000a0	W	0x00000000	Start address of source image(CR integer part)
IEP_SRC_ADDR_Y_F_TEMP	0x000a4	W	0x00000000	Start address of source image(Y fraction part)
IEP_SRC_ADDR_CBC_R_FTEMP	0x000a8	W	0x00000000	Start address of source image(CBCR fraction part)
IEP_SRC_ADDR_CR_FTEMP	0x000ac	W	0x00000000	Start address of source image(CR fraction part)
IEP_DST_ADDR_YRG_B	0x000b0	W	0x00000000	Start address of destination image(Y/RGB)
IEP_DST_ADDR_CBC_R	0x000b4	W	0x00000000	Start address of destination image(Cb/Cr)
IEP_DST_ADDR_CR	0x000b8	W	0x00000000	Start address of destination image(Cr)
IEP_DST_ADDR_Y1	0x000bc	W	0x00000000	Start address of destination image(Y)
IEP_DST_ADDR_CBC_R1	0x000c0	W	0x00000000	Start address of destination image(Cb/Cr)
IEP_DST_ADDR_CR1	0x000c4	W	0x00000000	Start address of destination image(Cr)
IEP_DST_ADDR_Y_I_TEMP	0x000c8	W	0x00000000	Start address of destination image(Y integer part)

Name	Offset	Size	Reset Value	Description
IEP_DST_ADDR_CBC_R_ITEMP	0x000cc	W	0x00000000	Start address of destination image(CBCR integer part)
IEP_DST_ADDR_CR_ITEMP	0x000d0	W	0x00000000	Start address of destination image(CR integer part)
IEP_DST_ADDR_Y_FTEMP	0x000d4	W	0x00000000	Start address of destination image(Y fraction part)
IEP_DST_ADDR_CBCR_FTEMP	0x000d8	W	0x00000000	Start address of destination image(CBCR fraction part)
IEP_DST_ADDR_CR_FTEMP	0x000dc	W	0x00000000	Start address of destination image(CR fraction part)
IEP_DIL_MTN_TAB0	0x000e0	W	0x00000000	Deinterlace motion table0
IEP_DIL_MTN_TAB1	0x000e4	W	0x00000000	Deinterlace motion table1
IEP_DIL_MTN_TAB2	0x000e8	W	0x00000000	Deinterlace motion table2
IEP_DIL_MTN_TAB3	0x000ec	W	0x00000000	Deinterlace motion table3
IEP_DIL_MTN_TAB4	0x000f0	W	0x00000000	Deinterlace motion table4
IEP_DIL_MTN_TAB5	0x000f4	W	0x00000000	Deinterlace motion table5
IEP_DIL_MTN_TAB6	0x000f8	W	0x00000000	Deinterlace motion table6
IEP_DIL_MTN_TAB7	0x000fc	W	0x00000000	Deinterlace motion table7
IEP_ENH(CG)_TAB	0x00100	W	0x00000000	contrast and gamma enhancement table
IEP_ENH(DDE)_COE0	0x00400	W	0x00000000	denoise,detail and edge enhancement coefficient
IEP_ENH(DDE)_COE1	0x00500	W	0x00000000	denoise,detail and edge enhancement coefficient
IEP_MMU_DTE_ADDR	0x00800	W	0x00000000	MMU current page table address
IEP_MMU_STATUS	0x00804	W	0x00000018	MMU status register
IEP_MMU_CMD	0x00808	W	0x00000000	MMU command register
IEP_MMU_PAGE_FAULT_ADDR	0x0080c	W	0x00000000	MMU logic address of last page fault

Name	Offset	Size	Reset Value	Description
IEP_MMU_ZAP_ONE_LINE	0x00810	W	0x00000000	MMU zap cache line register
IEP_MMU_INT_RAW_STAT	0x00814	W	0x00000000	MMU raw interrupt status register
IEP_MMU_INT_CLEA_R	0x00818	W	0x00000000	MMU interrupt clear register
IEP_MMU_INT_MASK	0x0081c	W	0x00000000	MMU interrupt mask register
IEP_MMU_INT_STATUS	0x00820	W	0x00000000	MMU interrupt status register
IEP_MMU_AUTO_GATING	0x00824	W	0x00000001	MMU clock auto gating register

Notes: **Size** : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

## 29.4.2 Detail Register Description

### IEP\_CONFIG

Address: Operational Base + offset (0x000000)  
configuration register0

Bit	Attr	Reset Value	Description
31:29	RO	0x0	reserved
28	RW	0x0	scl_en image scaling enable 0:disable 1:enable
27:26	RW	0x0	scl_sel Scaling select 00: horizontal down-scaling & vertical down-scaling; 01: horizontal down-scaling & vertical up-scaling; 10: horizontal up-scaling & vertical down-scaling; 11: horizontal up-scaling & vertical up-scaling;
25:24	RW	0x0	scl_up_coe_sel scale up coefficient select 00:hermite 01:spline 10:catrom 11:mitchell
23	RW	0x0	dil_ei_sel deinterlace edge interpolation select

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
22:21	RW	0x0	dil_ei_radius deinterlace edge interpolation radius
20	RW	0x0	rgb_con_gam_order RGB contrast enhancement and gamma adjustment operation order select. 0:CG prior to DDE 1:DDE prior to CG (CG represent for contrast & gamma operation, and DDE represent for denoise, detail or edge enhancement operation)
19:18	RW	0x0	rgb_enh_sel RGB enhancement select 00: no operation 01: denoise 10: detail enhancement 11: edge enhancement
17	RW	0x0	rgb_con_gam_en RGB contrast enhancement and gamma adjustment enable 0:disable 1:enable
16	RW	0x0	rgb_color_enh_en RGB color enhancement enable 0:disable 1:enable
15	RW	0x0	dil_ei_smooth deinterlace edge interpolation for smooth effect 0: disable 1: enable
14	RW	0x0	yuv_enh_en yuv enhancement enable 0:disable 1:enable
13	RW	0x0	yuv_dns_en YUV 3D denoise enable 0:disable 1:enable
12	RW	0x0	dil_ei_mode deinterlace edge interpolation 0: disable 1: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x0	dil_hf_en deinterlace high frequency calculation enable 0: disable 1: enable
10:8	RW	0x0	dil_mode Deinterlace mode select: 000: YUV deinterlace and bypass path disable; 001: I4O2 mode 010: I4O1B mode 011: I4O1T mode 100: I2O1B mode 101: I2O1T mode 110: bypass mode
7:1	RW	0x00	dil_hf_fct deinterlace high frequency factor
0	RW	0x0	vop_path_en VOP direct path enable 0:disable 1:enable

**IEP\_CONFIG1**

Address: Operational Base + offset (0x00004)

configuration register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	glb_alpha global alpha value only valid when destination format is ARGB
23	RW	0x0	rgb2yuv_input_clip RGB to YUV input range 0:R/G/B=[0,255] 1:R/G/B=[16,235]
22	RW	0x0	yuv2rgb_input_clip YUV to RGB input range 0:Y/U/V=[0,255] 1:Y=[16,235],U/V=[16,240]
21	RW	0x0	rgb_to_yuv_en RGB to YUV conversion enable 0: disable 1: enable
20	RW	0x0	yuv_to_rgb_en YUV to RGB conversion enable 0: disable 1: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19:18	RW	0x0	rgb2yuv_coe_sel rgb2yuv coefficient select 00:bt601_1 01:bt601_f 10:bt709_1 11:bt709_f
17:16	RW	0x0	yuv2rgb_coe_sel yuv2rgb coefficient select 00:bt601_1 01:bt601_f 10:bt709_1 11:bt709_f
15	RW	0x0	dthr_down_en dither down enable 0: disable 1: enable
14	RW	0x0	dthr_up_en dither up enable 0: disable 1: enable
13:12	RW	0x0	dst_yuv_swap destination YUV swap 00:SP UV 01:SP VU 10, 11:P
11:10	RW	0x0	dst_rgb_swap destination RGB swap ARGB destination 00:ARGB 01:ABGR 10:RGBA 11:BGRA RGB565 destination 00,10:RGB 01,11:BGR
9:8	RW	0x0	dst_fmt Output image Format 00 : ARGB 01 : RGB565 10 : YUV422 11 : YUV420
7:6	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:4	RW	0x0	src_yuv_swap source YUV swap 00:SP UV 01:SP VU 10, 11:P
3:2	RW	0x0	src_rgb_swap source RGB swap XRGB source 00:XRGB 01:XBGR 10:RGBX 11:BGRX RGB565 source 00,10:RGB 01,11:BGR
1:0	RW	0x0	src_fmt Input image Format 00 : XRGB 01 : RGB565 10 : YUV422 11 : YUV420

**IEP\_STATUS**

Address: Operational Base + offset (0x00008)

status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19	RW	0x0	rrgb_idle_ack RGB read DMA idle acknowlege
18	RW	0x0	wrgb_idle_ack RGB write DMA idle acknowlege
17	RW	0x0	ryuv_idle_ack YUV read DMA idle acknowlege
16	RW	0x0	wyuv_idle_ack YUV write DMA idle acknowlege
15:9	RO	0x0	reserved
8	RO	0x0	voi_sts vop direct path status 00:idle 01:working

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RO	0x0	rrgb_sts RGB DMA read status 00:idle 01:working
6	RO	0x0	wrgb_sts RGB DMA write status 00:idle 01:working
5	RO	0x0	ryuv_sts YUV DMA read status 00:idle 01:working
4	RO	0x0	wyuv_sts YUV DMA write status 00:idle 01:working
3	RO	0x0	dde_sts RGB denoise/enhancement status 00:idle 01:working
2	RO	0x0	dil_sts de-interlace or yuv bypass status 00:idle 01:working
1	RO	0x0	scl_sts scaling status 00:idle 01:working
0	RO	0x0	dns_sts YUV 3D denoise status 00:idle 01:working

**IEP\_INT**

Address: Operational Base + offset (0x0000c)  
 interrupt register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved
16	W1C	0x0	frm_done_int_clr Frame process done interrupt clear After be set to 1, this bit will be clear automatically.
15:9	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	RW	0x0	frm_done_int_en Frame process done interrupt enable: 0: disable; 1: enable;
7:1	RO	0x0	reserved
0	RO	0x0	frm_done_int Frame process done interrupt 0: inactive; 1: active;

**IEP\_FRM\_START**

Address: Operational Base + offset (0x00010)  
frame start

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	W1C	0x0	frm_start frame start Write 1, self clear.

**IEP\_CONFIG\_DONE**

Address: Operational Base + offset (0x00018)  
configuration done

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	config_done configuration done Wait for frame start to update raw register configuration to really used registers.

**IEP\_FRM\_CNT**

Address: Operational Base + offset (0x0001c)  
frame counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	frm_cnt frame counter Self increase one after a frame operation is finished. Write arbitrary value to clear to zero.

**IEP\_VIR\_IMG\_WIDTH**

Address: Operational Base + offset (0x00020)  
Image virtual width

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0140	dst_vir_image_width Destination virtual image width

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0140	src_vir_image_width Source virtual image width

**IEP\_IMG\_SCL\_FCT**

Address: Operational Base + offset (0x00024)

scaling factor

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x2000	vrt_scl_fct Vertical scale factor up scaling: $vrt\_scl\_fct=floor(src\_image\_height/dst\_image\_height);$ down scaling: $vrt\_scl\_fct=ceiling((dst\_image\_height+1)/(src\_image\_height+1));$
15:0	RW	0x2000	hrz_scl_fct Horizontal scale factor up scaling: $hrz\_scl\_fct=floor(src\_image\_width/dst\_image\_width);$ down scaling: $hrz\_scl\_fct=ceiling((dst\_image\_width+1)/(src\_image\_width+1));$

**IEP\_SRC\_IMG\_SIZE**

Address: Operational Base + offset (0x00028)

Source image width/height

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x00f0	src_image_height source image height
15:13	RO	0x0	reserved
12:0	RW	0x0140	src_image_width source image width

**IEP\_DST\_IMG\_SIZE**

Address: Operational Base + offset (0x0002c)

Destination image width/height

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
28:16	RW	0x00f0	dst_image_height Destination image height
15:13	RO	0x0	reserved
12:0	RW	0x0140	dst_image_width Destination image width

**IEP\_DST\_IMG\_WIDTH\_TILE0**

Address: Operational Base + offset (0x00030)

Destination image tile0 width

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	dst_image_width_tile0 Destination image tile0 width

**IEP\_DST\_IMG\_WIDTH\_TILE1**

Address: Operational Base + offset (0x00034)

Destination image tile1 width

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved
10:0	RW	0x000	dst_image_width_tile1 Destination image tile1 width

**IEP\_DST\_IMG\_WIDTH\_TILE2**

Address: Operational Base + offset (0x00038)

Destination image tile2 width

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved
10:0	RW	0x000	dst_image_width_tile2 Destination image tile2 width

**IEP\_DST\_IMG\_WIDTH\_TILE3**

Address: Operational Base + offset (0x0003c)

Destination image tile3 width

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved
10:0	RW	0x000	dst_image_width_tile3 Destination image tile3 width

**IEP\_ENH\_YUV\_CNFG\_0**

Address: Operational Base + offset (0x00040)

brightness,contrast,saturation adjustment

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:16	RW	0x000	sat_con YUV saturation and contrast adjustment saturation * contrast range from 0 to 1.992*1.992, and this value is saturation* contrast * 128
15:8	RW	0x00	contrast YUV contrast adjustment contrast value range from 0 to 1.992, and this value is contrast*128.
7:6	RO	0x0	reserved
5:0	RW	0x00	brightness YUV brightness adjustment range from -32 to 31 000000:0; 000001:1; ..... 011111:31; 100000:-32; 100001:-31; ..... 111110:-2; 111111:-1;

**IEP\_ENH\_YUV\_CNFG\_1**

Address: Operational Base + offset (0x00044)

Hue configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:8	RW	0x00	cos_hue the cos function value for hue adjustment sin function value range from 0.866 to 1 ,and this value is cos * 128 ,no sign bit
7:0	RW	0x00	sin_hue the sin function value for hue adjustment sin function value range from -0.5 to 0.5 ,and this value is sin * 128 ,and the high bit is sign bit

**IEP\_ENH\_YUV\_CNFG\_2**

Address: Operational Base + offset (0x00048)

color bar configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RO	0x0	reserved
25:24	RW	0x0	video_mode video mode 00:black screen 01:blue screen 10:color bars 11:normal video
23:16	RW	0x00	color_bar_v color bar v value
15:8	RW	0x00	color_bar_u color bar u value
7:0	RW	0x00	color_bar_y color bar y value

**IEP\_ENH\_RGB\_CNFG**

Address: Operational Base + offset (0x0004c)

enhancement RGB configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RW	0x0	luma_spat_sel 3D denoise luma spatial coefficient select
29:28	RW	0x0	luma_temp_sel 3D denoise luma temporal coefficient select
27:26	RW	0x0	chroma_spat_sel 3D denoise chroma spatial coefficient select
25:24	RW	0x0	chroma_temp_sel 3D denoise chroma temporal coefficient select
23:16	RW	0x00	enh_threshold enhancement threshold In denoise and detail enhancement operation, more than the threshold, considering as detail; but if less than the threshold, considering as noise, need to be filtered.
15	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14:8	RW	0x00	<p>enh_alpha enhancement alpha value 0000000:0 0000001:1/16 0000010:2/16 ..... 0001111:15/16 0010000:1 0010001:1+1/16; 0010010:1+2/16; 0010011:1+3/16; ..... 0100000:2; ..... 0110000:3; ..... 1000000:4; ..... 1010000:5; ..... 1100000:6; other : reserved</p>
7:2	RO	0x0	reserved
1:0	RW	0x0	<p>enh_radius enhancement radius 00:R=1 01:R=2 10:R=3 11:R=4</p>

**IEP\_ENH\_C\_COE**

Address: Operational Base + offset (0x00050)

rgb color enhancement coefficient

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6:5	RW	0x0	c_int_coe color enhancement integer coefficient
4:0	RW	0x00	c_frac_coe color enhancement fraction coefficient

**IEP\_RAW\_CONFIG0**

Address: Operational Base + offset (0x00058)

configuration register0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	vrt_inv vertical inverse display 0: normal display 1: inverse display
30	RW	0x0	hrz_inv horizontal inverse display 0: normal display 1: inverse display
29	RO	0x0	reserved
28	RW	0x0	scl_en image scaling enable 0: disable 1: enable
27:26	RW	0x0	scl_sel Scaling select 00: horizontal down-scaling & vertical down-scaling; 01: horizontal down-scaling & vertical up-scaling; 10: horizontal up-scaling & vertical down-scaling; 11: horizontal up-scaling & vertical up-scaling;
25:24	RW	0x0	scl_up_coe_sel scale up coefficient select 00: hermite 01: spline 10: catrom 11: mitchell
23	RW	0x0	dil_ei_sel deinterlace edge interpolation select
22:21	RW	0x0	dil_ei_radius deinterlace edge interpolation radius
20	RW	0x0	rgb_con_gam_order RGB contrast enhancement and gamma adjustment operation order select. 0: CG prior to DDE 1: DDE prior to CG (CG represent for contrast & gamma operation, and DDE represent for denoise, detail or edge enhancement operation)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19:18	RW	0x0	rgb_enh_sel RGB enhancement select 00: no operation 01: denoise 10: detail enhancement 11: edge enhancement
17	RW	0x0	rgb_con_gam_en RGB contrast enhancement and gamma adjustment enable 0:disable 1:enable
16	RW	0x0	rgb_color_enh_en RGB color enhancement enable 0:disable 1:enable
15	RW	0x0	dil_ei_smooth deinterlace edge interpolation for smooth effect 0: disable 1: enable
14	RW	0x0	yuv_enh_en yuv enhancement enable 0:disable 1:enable
13	RW	0x0	yuv_dns_en YUV 3D denoise enable 0:disable 1:enable
12	RW	0x0	dil_ei_mode deinterlace edge interpolation 0: disable 1: enable
11	RW	0x0	dil_hf_en deinterlace high frequency calculation enable 0: disable 1: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
10:8	RW	0x0	dil_mode Deinterlace mode select: 000: YUV deinterlace and bypass path disable; 001: I4O2 mode 010: I4O1B mode 011: I4O1T mode 100: I2O1B mode 101: I2O1T mode 110: bypass mode
7:1	RW	0x00	dil_hf_fct deinterlace high frequency factor
0	RW	0x0	vop_path_en VOP direct path enable 0:disable 1:enable

**IEP\_RAW\_CONFIG1**

Address: Operational Base + offset (0x0005c)

configuration register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x00	glb_alpha global alpha value only valid when destination format is ARGB
23	RO	0x0	rgb2yuv_input_clip RGB to YUV input range 0:R/G/B=[0,255] 1:R/G/B=[16,235]
22	RO	0x0	yuv2rgb_input_clip YUV to RGB input range 0:Y/U/V=[0,255] 1:Y=[16,235],U/V=[16,240]
21	RO	0x0	rgb_to_yuv_en RGB to YUV conversion enable 0: disable 1: enable
20	RO	0x0	yuv_to_rgb_en YUV to RGB conversion enable 0: disable 1: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19:18	RO	0x0	rgb2yuv_coe_sel rgb2yuv coefficient select 00:bt601_1 01:bt601_f 10:bt709_1 11:bt709_f
17:16	RO	0x0	yuv2rgb_coe_sel yuv2rgb coefficient select 00:bt601_1 01:bt601_f 10:bt709_1 11:bt709_f
15	RO	0x0	dthr_down_en dither down enable 0: disable 1: enable
14	RO	0x0	dthr_up_en dither up enable 0: disable 1: enable
13:12	RO	0x0	dst_yuv_swap destination YUV swap 00:SP UV 01:SP VU 10, 11:P
11:10	RO	0x0	dst_rgb_swap destination RGB swap ARGB destination 00:ARGB 01:ABGR 10:RGBA 11:BGRA RGB565 destination 00,10:RGB 01,11:BGR
9:8	RO	0x0	dst_fmt Output image Format 00 : ARGB 01 : RGB565 10 : YUV422 11 : YUV420
7:6	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:4	RO	0x0	src_yuv_swap source YUV swap 00:SP UV 01:SP VU 10, 11:P
3:2	RO	0x0	src_rgb_swap source RGB swap XRGB source 00:XRGB 01:XBGR 10:RGBX 11:BGRX RGB565 source 00,10:RGB 01,11:BGR
1:0	RO	0x0	src_fmt Input image Format 00 : XRGB 01 : RGB565 10 : YUV422 11 : YUV420

**IEP\_RAW\_VIR\_IMG\_WIDTH**

Address: Operational Base + offset (0x00060)

Image virtual width

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0140	dst_vir_image_width Destination virtual image width
15:0	RO	0x0140	src_vir_image_width Source virtual image width

**IEP\_RAW\_IMG\_SCL\_FCT**

Address: Operational Base + offset (0x00064)

scaling factor

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x2000	<p>vrt_scl_fct Vertical scale factor up scaling: <math>vrt\_scl\_fct=floor(src\_image\_height/dst\_image\_height);</math> down scaling: <math>vrt\_scl\_fct=ceiling((dst\_image\_height+1)/(src\_image\_height+1));</math></p>
15:0	RO	0x2000	<p>hrz_scl_fct Horizontal scale factor up scaling: <math>hrz\_scl\_fct=floor(src\_image\_width/dst\_image\_width);</math> down scaling: <math>hrz\_scl\_fct=ceiling((dst\_image\_width+1)/(src\_image\_width+1));</math></p>

**IEP\_RAW\_SRC\_IMG\_SIZE**

Address: Operational Base + offset (0x00068)

Source image width/height

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RO	0x00f0	src_image_height source image height
15:13	RO	0x0	reserved
12:0	RO	0x0140	src_image_width source image width

**IEP\_RAW\_DST\_IMG\_SIZE**

Address: Operational Base + offset (0x0006c)

Destination image width/height

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RO	0x00f0	dst_image_height Destination image height
15:13	RO	0x0	reserved
12:0	RO	0x0140	dst_image_width Destination image width

**IEP\_RAW\_ENH\_YUV\_CNFG\_0**

Address: Operational Base + offset (0x00070)

brightness,contrast,saturation adjustment

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24:16	RO	0x000	sat_con YUV saturation and contrast adjustment saturation * contrast range from 0 to 1.992*1.992, and this value is saturation* contrast * 128
15:8	RO	0x00	contrast YUV contrast adjustment contrast value range from 0 to 1.992, and this value is contrast*128.
7:6	RO	0x0	reserved
5:0	RO	0x00	brightness YUV brightness adjustment range from -32 to 31 000000:0; 000001:1; ..... 011111:31; 100000:-32; 100001:-31; ..... 111110:-2; 111111:-1;

### **IEP\_RAW\_ENH\_YUV\_CNFG\_1**

Address: Operational Base + offset (0x00074)

Hue configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:8	RO	0x00	cos_hue the cos function value for hue adjustment sin function value range from 0.866 to 1 ,and this value is cos * 128 ,no sign bit
7:0	RO	0x00	sin_hue the sin function value for hue adjustment sin function value range from -0.5 to 0.5 ,and this value is sin * 128 ,and the high bit is sign bit

### **IEP\_RAW\_ENH\_YUV\_CNFG\_2**

Address: Operational Base + offset (0x00078)

color bar configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
25:24	RO	0x0	video_mode video mode 00:black screen 01:blue screen 10:color bars 11:normal video
23:16	RO	0x00	color_bar_v color bar v value
15:8	RO	0x00	color_bar_u color bar u value
7:0	RO	0x00	color_bar_y color bar y value

**IEP\_RAW\_ENH\_RGB\_CNFG**

Address: Operational Base + offset (0x0007c)

enhancement RGB configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RW	0x0	luma_spat_sel 3D denoise luma spatial coefficient select
29:28	RW	0x0	luma_temp_sel 3D denoise luma temporal coefficient select
27:26	RW	0x0	chroma_spat_sel 3D denoise chroma spatial coefficient select
25:24	RW	0x0	chroma_temp_sel 3D denoise chroma temporal coefficient select
23:16	RW	0x00	enh_threshold enhancement threshold In denoise and detail enhancement operation, more than the threshold, considering as detail; but if less than the threshold, considering as noise, need to be filtered.
15	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14:8	RW	0x00	enh_alpha enhancement alpha value 0000000:0 0000001:1/16 0000010:2/16 ..... 0001111:15/16 0010000:1 0010001:1+1/16; 0010010:1+2/16; 0010011:1+3/16; ..... 0100000:2; ..... 0110000:3; ..... 1000000:4; ..... 1010000:5; ..... 1100000:6; other : reserved
7:2	RO	0x0	reserved
1:0	RW	0x0	enh_radius enhancement radius 00:R=1 01:R=2 10:R=3 11:R=4

**IEP\_SRC\_ADDR\_YRGB**

Address: Operational Base + offset (0x00080)

Start address of source image(Y/RGB)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_yrgb_mst Source image data YRGB start address in Memory

**IEP\_SRC\_ADDR\_CBCR**

Address: Operational Base + offset (0x00084)

Start address of source image(Cb/Cr)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_cbcr_mst Source image data CbCr start address in Memory

**IEP\_SRC\_ADDR\_CR**

Address: Operational Base + offset (0x00088)

Start address of source image(Cr)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_cr_mst Source image data Cr start address in Memory

**IEP\_SRC\_ADDR\_Y1**

Address: Operational Base + offset (0x0008c)

Start address of source image(Y)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_y_mst Source image data Y start address in Memory

**IEP\_SRC\_ADDR\_CBCR1**

Address: Operational Base + offset (0x00090)

Start address of source image(Cb/Cr)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_cbcr_mst Source image data CbCr start address in Memory

**IEP\_SRC\_ADDR\_CR1**

Address: Operational Base + offset (0x00094)

Start address of source image(Cr)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_cr_mst Source image data Cr start address in Memory

**IEP\_SRC\_ADDR\_Y\_ITEMP**

Address: Operational Base + offset (0x00098)

Start address of source image(Y integer part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_y_mst_itemp Interger part source image data Y start address in Memory

**IEP\_SRC\_ADDR\_CBCR\_ITEMP**

Address: Operational Base + offset (0x0009c)  
Start address of source image(CBCR integer part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_cbcr_mst_cbcr_itemp Interger part source image data CBCR start address in Memory

**IEP\_SRC\_ADDR\_CR\_ITEMP**

Address: Operational Base + offset (0x000a0)  
Start address of source image(CR integer part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_cr_mst_cr_itemp Interger part source image data CR start address in Memory

**IEP\_SRC\_ADDR\_Y\_FTEMP**

Address: Operational Base + offset (0x000a4)  
Start address of source image(Y fraction part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_y_mst_ftemp Fraction part source image data Y start address in Memory

**IEP\_SRC\_ADDR\_CBCR\_FTEMP**

Address: Operational Base + offset (0x000a8)  
Start address of source image(CBCR fraction part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_cbcr_mst_ftemp Fraction part source image data CBCR start address in Memory

**IEP\_SRC\_ADDR\_CR\_FTEMP**

Address: Operational Base + offset (0x000ac)  
Start address of source image(CR fraction part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	src_image_cr_mst_ftemp Fraction part source image data CR start address in Memory

**IEP\_DST\_ADDR\_YRGB**

Address: Operational Base + offset (0x000b0)

Start address of destination image(Y/RGB)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_yrgb_mst Destination image data YRGB start address in Memory

**IEP\_DST\_ADDR\_CBCR**

Address: Operational Base + offset (0x000b4)

Start address of destination image(Cb/Cr)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_cbcn_mst Destination image data CBCR start address in Memory

**IEP\_DST\_ADDR\_CR**

Address: Operational Base + offset (0x000b8)

Start address of destination image(Cr)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_cr_mst Destination image data CR start address in Memory

**IEP\_DST\_ADDR\_Y1**

Address: Operational Base + offset (0x000bc)

Start address of destination image(Y)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_y_mst Destination image data Y start address in Memory

**IEP\_DST\_ADDR\_CBCR1**

Address: Operational Base + offset (0x000c0)

Start address of destination image(Cb/Cr)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_cbcr_mst Destination image data CbCr start address in Memory

**IEP\_DST\_ADDR\_CR1**

Address: Operational Base + offset (0x000c4)

Start address of destination image(Cr)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_cr_mst Destination image data Cr start address in Memory

**IEP\_DST\_ADDR\_Y\_ITEMP**

Address: Operational Base + offset (0x000c8)

Start address of destination image(Y integer part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_y_mst_itemp Intger part destination image data Y start address in Memory

**IEP\_DST\_ADDR\_CBCR\_ITEMP**

Address: Operational Base + offset (0x000cc)

Start address of destination image(CBCR integer part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_cbcr_mst_itemp Int part destination image data CBCR start address in Memory

**IEP\_DST\_ADDR\_CR\_ITEMP**

Address: Operational Base + offset (0x000d0)

Start address of destination image(CR integer part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_cr_mst_itemp Intger part destination image data CR start address in Memory

**IEP\_DST\_ADDR\_Y\_FTEMP**

Address: Operational Base + offset (0x000d4)

Start address of destination image(Y fraction part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_y_mst_ftemp Fraction part destination image data Y start address in Memory

**IEP\_DST\_ADDR\_CBCR\_FTEMP**

Address: Operational Base + offset (0x000d8)

Start address of destination image(CBCR fraction part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_cbcr_mst_ftemp Fraction part destination image data CBCR start address in Mem

**IEP\_DST\_ADDR\_CR\_FTEMP**

Address: Operational Base + offset (0x000dc)

Start address of destination image(CR fraction part)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	dst_image_cr_mst_ftemp Fraction part destination image data CR start address

**IEP\_DIL\_MTN\_TAB0**

Address: Operational Base + offset (0x000e0)

Deinterlace motion table0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved
30:24	RW	0x00	mtn_sub_tab3 motion sub table3
23	RO	0x0	reserved
22:16	RW	0x00	mtn_sub_tab2 motion sub table2
15	RO	0x0	reserved
14:8	RW	0x00	mtn_sub_tab1 motion sub table1
7	RO	0x0	reserved
6:0	RW	0x00	mtn_sub_tab0 motion sub table0

**IEP\_DIL\_MTN\_TAB1**

Address: Operational Base + offset (0x000e4)

Deinterlace motion table1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved
30:24	RW	0x00	mtn_sub_tab3 motion sub table3
23	RO	0x0	reserved
22:16	RW	0x00	mtn_sub_tab2 motion sub table2
15	RO	0x0	reserved
14:8	RW	0x00	mtn_sub_tab1 motion sub table1
7	RO	0x0	reserved
6:0	RW	0x00	mtn_sub_tab0 motion sub table0

**IEP\_DIL\_MTN\_TAB2**

Address: Operational Base + offset (0x000e8)

Deinterlace motion table2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved
30:24	RW	0x00	mtn_sub_tab3 motion sub table3
23	RO	0x0	reserved
22:16	RW	0x00	mtn_sub_tab2 motion sub table2
15	RO	0x0	reserved
14:8	RW	0x00	mtn_sub_tab1 motion sub table1
7	RO	0x0	reserved
6:0	RW	0x00	mtn_sub_tab0 motion sub table0

**IEP\_DIL\_MTN\_TAB3**

Address: Operational Base + offset (0x000ec)

Deinterlace motion table3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved
30:24	RW	0x00	mtn_sub_tab3 motion sub table3
23	RO	0x0	reserved
22:16	RW	0x00	mtn_sub_tab2 motion sub table2
15	RO	0x0	reserved
14:8	RW	0x00	mtn_sub_tab1 motion sub table1
7	RO	0x0	reserved
6:0	RW	0x00	mtn_sub_tab0 motion sub table0

**IEP\_DIL\_MTN\_TAB4**

Address: Operational Base + offset (0x000f0)

Deinterlace motion table4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved
30:24	RW	0x00	mtn_sub_tab3 motion sub table3
23	RO	0x0	reserved
22:16	RW	0x00	mtn_sub_tab2 motion sub table2
15	RO	0x0	reserved
14:8	RW	0x00	mtn_sub_tab1 motion sub table1
7	RO	0x0	reserved
6:0	RW	0x00	mtn_sub_tab0 motion sub table0

**IEP\_DIL\_MTN\_TAB5**

Address: Operational Base + offset (0x000f4)

Deinterlace motion table5

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
30:24	RW	0x00	mtn_sub_tab3 motion sub table3
23	RO	0x0	reserved
22:16	RW	0x00	mtn_sub_tab2 motion sub table2
15	RO	0x0	reserved
14:8	RW	0x00	mtn_sub_tab1 motion sub table1
7	RO	0x0	reserved
6:0	RW	0x00	mtn_sub_tab0 motion sub table0

**IEP\_DIL\_MTN\_TAB6**

Address: Operational Base + offset (0x000f8)

Deinterlace motion table6

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved
30:24	RW	0x00	mtn_sub_tab3 motion sub table3
23	RO	0x0	reserved
22:16	RW	0x00	mtn_sub_tab2 motion sub table2
15	RO	0x0	reserved
14:8	RW	0x00	mtn_sub_tab1 motion sub table1
7	RO	0x0	reserved
6:0	RW	0x00	mtn_sub_tab0 motion sub table0

**IEP\_DIL\_MTN\_TAB7**

Address: Operational Base + offset (0x000fc)

Deinterlace motion table7

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RO	0x0	reserved
30:24	RW	0x00	mtn_sub_tab3 motion sub table3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23	RO	0x0	reserved
22:16	RW	0x00	mtn_sub_tab2 motion sub table2
15	RO	0x0	reserved
14:8	RW	0x00	mtn_sub_tab1 motion sub table1
7	RO	0x0	reserved
6:0	RW	0x00	mtn_sub_tab0 motion sub table0

**IEP\_ENH(CG)\_TAB**

Address: Operational Base + offset (0x00100)

contrast and gamma enhancement table

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	cg_tab_3 cg table 3 pixel value 3,7,11,15,.....mapping
23:16	RW	0x00	cg_tab_2 cg table 2 pixel value 2,6,10,14,.....mapping
15:8	RW	0x00	cg_tab_1 cg table 1 pixel value 1,5,9,13,.....mapping
7:0	RW	0x00	cg_tab_0 cg table 0 256x8bit contrast & gamma mapping table pixel value 0,4,8,12,.....mapping

**IEP\_ENH(DDE)\_COEO**

Address: Operational Base + offset (0x00400)

denoise,detail and edge enhancement coefficient

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved
29:24	RW	0x00	dde_coe_3 dde coefficient 3 coefficient number 3,7,11,15,.....
23:22	RO	0x0	reserved
21:16	RW	0x00	dde_coe_2 dde coefficient 2 coefficient number 2,6,10,14,.....
15:14	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13:8	RW	0x00	dde_coe_1 dde coefficient 1 coefficient number 1,5,9,13,.....
7:6	RO	0x0	reserved
5:0	RW	0x00	dde_coe_0 dde coefficient 0 256x6bit coefficient for denoise and detail enhancement coefficient number 0,4,8,12,.....

**IEP\_ENH\_DDE\_COE1**

Address: Operational Base + offset (0x00500)  
denoise,detail and edge enhancement coefficient

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved
29:24	RW	0x00	dde_coe_3 dde coefficient 3 coefficient number 3,7,11,15,.....
23:22	RO	0x0	reserved
21:16	RW	0x00	dde_coe_2 dde coefficient 3 coefficient number 2,6,10,14,.....
15:14	RO	0x0	reserved
13:8	RW	0x00	dde_coe_1 dde coefficient 1 coefficient number 1,5,9,13,.....
7:6	RO	0x0	reserved
5:0	RW	0x00	dde_coe_0 dde coefficient 1 81x6bit coefficient for denoise and detail enhancement coefficient number 0,4,8,12,.....

**IEP\_MMU\_DTE\_ADDR**

Address: Operational Base + offset (0x00800)  
MMU current page table address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	mmu_dte_addr page table address

**IEP\_MMU\_STATUS**

Address: Operational Base + offset (0x00804)  
MMU status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved
10:6	RW	0x00	mmu_page_fault_bus_id Index of master responsible for the last page fault
5	RW	0x0	mmu_page_fault_is_write The direction of access for last page fault: 0: read 1:write
4	RW	0x1	mmu_replay_buffer_empty The MMU replay buffer is empty.
3	RW	0x1	mmu_idle the MMU is idle when accesses are being translated and there are no unfinished translated access. The MMU_IDLE signal only reports idle when the MMU processor is idle and accesses are active on the external bus. Note: the MMU can be idle in page fault mode.
2	RW	0x0	mmu_stall_active MMU stall mode currently enabled. The mode is enabled by command.
1	RW	0x0	mmu_page_fault_active MMU page fault mode currently enabled. The mode is enabled by command
0	RW	0x0	mmu_paging_enabled mmu paging is enabled

**IEP\_MMU\_CMD**

Address: Operational Base + offset (0x00808)

MMU command register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:0	RW	0x0	<p>mmu_cmd</p> <p>0: MMU_ENABLE_PAGING. enable paging.</p> <p>1: MMU_DISABLE_PAGING. disable paging.</p> <p>2: MMU_ENABLE_STALL. turn on stall mode.</p> <p>3: MMU_DISABLE_STALL. turn off stall mode.</p> <p>4: MMU_ZAP_CACHE. zap the entire page table cache.</p> <p>5: MMU_PAGE_FAULT_DONE. leave page fault mode.</p> <p>6: MMU_FORCE_RESET. reset the mmu.</p> <p>The MMU_ENABLE_STALL command can always be issued.</p> <p>Other commands are ignored unless the MMU is idle or stalled.</p>

**IEP\_MMU\_PAGE\_FAULT\_ADDR**

Address: Operational Base + offset (0x0080c)

MMU logic address of last page fault

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	mmu_page_fault_addr address of last page fault

**IEP\_MMU\_ZAP\_ONE\_LINE**

Address: Operational Base + offset (0x00810)

MMU zap cache line register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	mmu_zap_one_line address to be invalidated from the page table cache.

**IEP\_MMU\_INT\_RAWSTAT**

Address: Operational Base + offset (0x00814)

MMU raw interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	read_bus_error read bus error
0	RW	0x0	page_fault page fault

**IEP\_MMU\_INT\_CLEAR**

Address: Operational Base + offset (0x00818)

MMU interrupt clear register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	read_bus_error_clear read bus error interrupt clear. write 1 to this register can clear read bus error interrupt.
0	RW	0x0	page_fault_clear page fault interrupt clear, write 1 to this register can clear page fault interrupt.

**IEP\_MMU\_INT\_MASK**

Address: Operational Base + offset (0x0081c)

MMU interrupt mask register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	read_bus_error_int_en read bus error interrupt enable
0	RW	0x0	page_fault_int_en page fault interrupt enable

**IEP\_MMU\_INT\_STATUS**

Address: Operational Base + offset (0x00820)

MMU interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	read_bus_error read bus error interrupt
0	RW	0x0	page_fault page fault interrupt

**IEP\_MMU\_AUTO\_GATING**

Address: Operational Base + offset (0x00824)

MMU clock auto gating register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x1	mmu_auto_gating mmu clock auto gating when it is 1, the mmu will auto gating itself

## 29.5 Application Notes

### 29.5.1 VOP path disabled configure flow

1. Keep IEP direct path disabled.
2. Configure all registers which are needed at any time.
3. Configure IEP\_CONFIG\_DONE.
4. Configure IEP\_FRM\_START.

### 29.5.2 VOP path enabled configure flow

1. Keep IEP direct path enabled.
2. Configure all IEP registers which are needed.
3. Configure VOP related registers which are needed.
4. Configure CONFIG\_DONE register in VOP only.
5. Wait for frame start from VOP and IEP direct path.

### 29.5.3 VOP path turn on flow

1. Configure all IEP registers which are needed.
2. Configure VOP related registers which are needed.
3. Enable IEP direct path.
4. Enable VOP direct path.
5. Configure CONFIG\_DONE register in VOP only.
6. Wait for frame start from VOP and IEP direct path.

### 29.5.4 VOP path turn off flow

1. Disable VOP direct path.
2. Disable IEP direct path, so IEP do not receive any other CONFIG\_DONE and frame start from VOP immediately.
3. Configure CONFIG\_DONE register in VOP.
4. Wait for frame start from VOP and IEP direct path, so VOP quit direct path mode completely.
5. Configure IEP registers which are needed at any time.
6. Configure IEP\_CONFIG\_DONE.
7. Configure IEP\_FRM\_START, IEP is working at write back mode now.

## Chapter 30 Video Input Processor (VIP)

### 30.1 Overview

The Video Input Processor, receives the data from Camera or CCIR656 encoder, and transfers the data into system main memory by AXI bus.

The Video Input Processor supports following features:

- Support YCbCr422 input
- Support Raw8bit input
- Support CCIR656(PAL/NTSC) input
- Support JPEG input
- Support YCbCr422/420 output
- Support UYVY/VYUY/YUYV/YVYU configurable
- Support up to 8192x8192 resolution source
- Support picture in picture
- Support arbitrary size window crop
- Support error/terminate interrupt and combined interrupt output
- Support clk/vsync/href polarity configurable
- Support one frame stop/ping-pong mode

### 30.2 Block Diagram

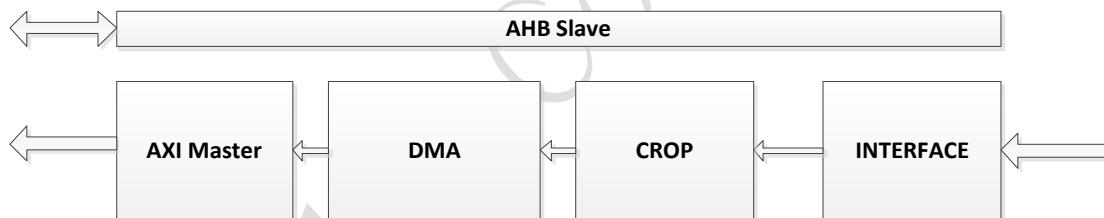


Fig. 30-1 VIP block diagram

The VIP comprises with:

- AHB Slave  
Host configure the registers via the AHB Slave
- AXI Master  
Transmit the data to chip memory via the AXI Master
- INTERFACE  
Translate the input video data into the requisite data format
- CROP  
Bypass or crop the source video data to a smaller size destination
- DMA  
Control the operation of AXI Master

### 30.3 Function description

This chapter is used to illustrate the operational behavior of how VIP works. If YUV422 or ccir656 signal is received from external devices, VIP translate it into YUV422/420 data, and separate the data to Y and UV data, then store them to different memory via AXI bus separately. But if raw data is received, there are not any translations happened, the 8 data is considered as 16bit data and write directly to memory.

### 30.3.1 Support Vsync high active or low active

- Vsync Low active as below

Vertical sensor timing (line by line)

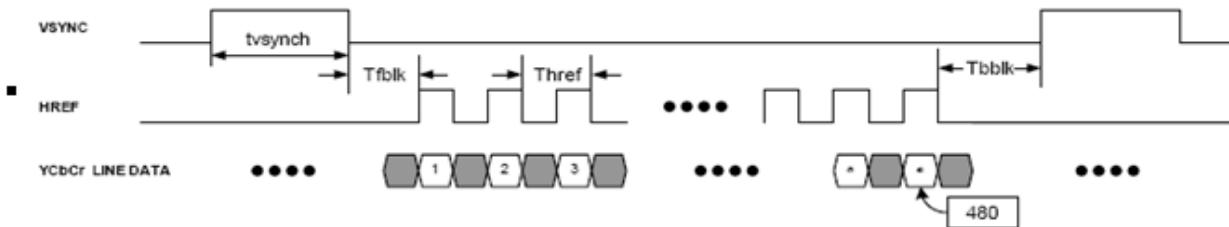


Fig. 30-2 Timing diagram for VIP when vsync low active

- Vsync High active

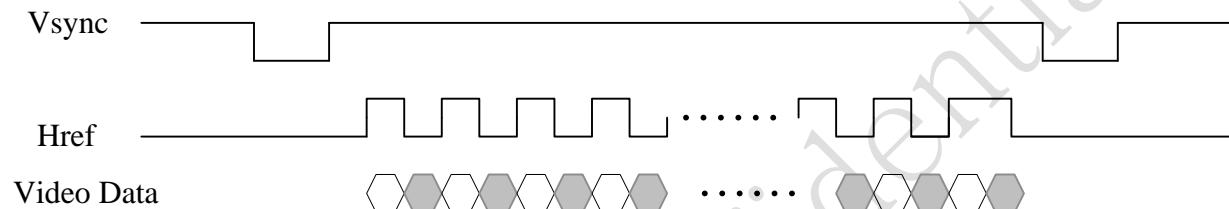


Fig. 30-3 Timing diagram for VIP when vsync high active

### 30.3.2 Support href high active or low active

- Href high active

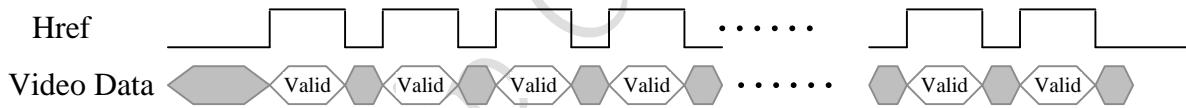


Fig. 30-4 Timing diagram for VIP when href high active

- Href Low active

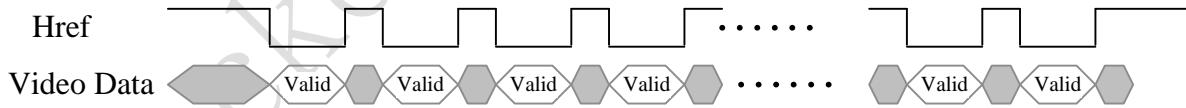


Fig. 30-5 Timing diagram for VIP when href low active

- Y first

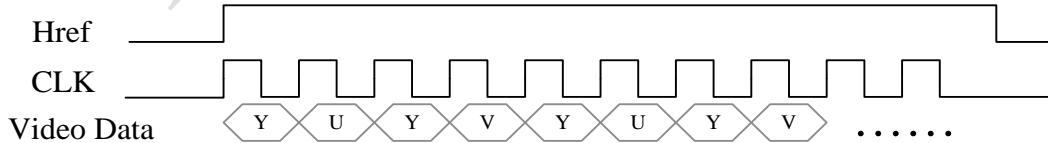


Fig. 30-6 Timing diagram for VIP when Y data first

- U first

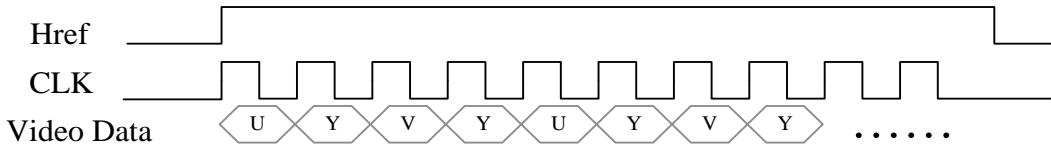


Fig. 30-7 Timing diagram for VIP when U data first

### 30.3.3 Support CCIR656 (NTSC and PAL)

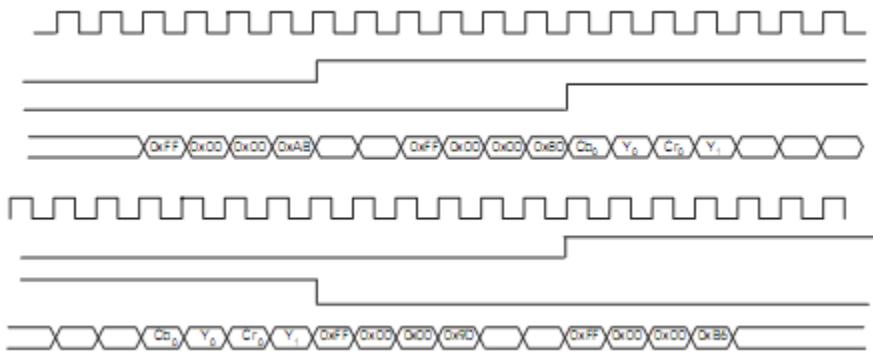


Fig. 30-8 CCIR656 timing

### 30.3.4 Support Raw data (8-bit) or JPEG

#### Pixel Data Timing Example

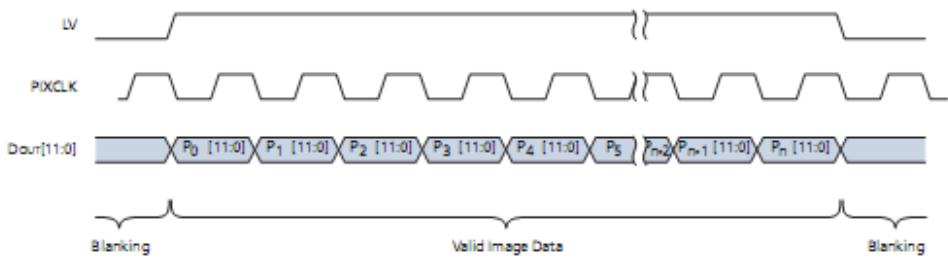


Fig. 30-9 Raw Data or JPEG Timing

VIP module can work in three modes: one frame stop mode, ping-pong mode.

#### One frame stop mode

In this mode, configure the parameter WORK\_MODE to one frame stop mode. After one frame captured, VIP will automatic stop. After capturing, the image Y, UV data will be stored at main memory location defined by VIP\_FRM0\_ADDR\_Y, FRM0\_ADDR\_UV separately.

#### Ping-Pong mode

After one frame(F1) captured, VIP will start to capture the next frame(F2) automatically, and host must assign new address pointer of frame1 and clear the frame1 status, thus VIP will capture the third frame automatically(by new F1 address) without any stop and so on for the following frames. But if host did not update the frame buffer address, the VIP will cover the pre-frame data stored in the memory with the following frame data.

#### Storage

Difference between the YUV mode and raw mode is that in the YUV mode or ccir656 mode, data will be storage in the Y data buffer and UV data buffer; but in the raw or jpeg mode, RGB data will be storage in the same buffer. In addition, in the yuv mode, the width of Y, U or V data

is a byte in memory; in Raw or JPEG mode, the width is a halfword no matter the data source is 8 bit.

## CROP

The parameter START\_Y and START\_X defines the coordinate of crop start point. And the frame size after cropping is following the value of SET\_WIDTH and SET\_HEIGHT.

## 30.4 Register description

### 30.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
VIP_CTRL	0x00000	W	0x00007000	VIP control
VIP_INTEN	0x00004	W	0x00000000	VIP interrupt status
VIP_INTSTAT	0x00008	W	0x00000000	VIP interrupt status
VIP_FOR	0x0000c	W	0x00000000	VIP format
VIP_FRM0_ADDR_Y	0x00014	W	0x00000000	VIP frame0 y address
VIP_FRM0_ADDR_UV	0x00018	W	0x00000000	VIP frame0 uv address
VIP_FRM1_ADDR_Y	0x0001c	W	0x00000000	VIP frame1 y address
VIP_FRM1_ADDR_UV	0x00020	W	0x00000000	VIP frame1 uv address
VIP_VIR_LINE_WIDT H	0x00024	W	0x00000000	VIP virtual line width
VIP_SET_SIZE	0x00028	W	0x01e002d0	VIP frame set size
VIP_CROP	0x00044	W	0x00000000	VIP crop start point
VIP_SCL_CTRL	0x00048	W	0x00000000	VIP scale control
VIP_FIFO_ENTRY	0x00054	W	0x00000000	VIP FIFO entry
VIP_FRAME_STATUS	0x00060	W	0x00000000	VIP frame status
VIP_CUR_DST	0x00064	W	0x00000000	VIP current destination address
VIP_LAST_LINE	0x00068	W	0x00000000	VIP last frame line number
VIP_LAST_PIX	0x0006c	W	0x00000000	VIP last line pixel number

Notes: *Size* : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

### 30.4.2 Detail Register Description

#### VIP\_CTRL

Address: Operational Base + offset (0x00000)

VIP control

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:12	RW	0x7	AXI_BURST_TYPE axi master burst type 0-15 : burst1~16
11:3	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:1	RW	0x0	WORK_MODE Working Mode 00 : one frame stop mode 01 : ping-pong mode 02 : line loop mode 03 : reserved
0	RW	0x0	CAP_EN capture enable 0 : disable 1 : enable

**VIP\_INTEN**

Address: Operational Base + offset (0x00004)

VIP interrupt status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9	RW	0x0	PST_INF_FRAME_END_EN frame end after interface FIFO interrupt enable 0 : disable 1 : enable
8	RW	0x0	PRE_INF_FRAME_END_EN frame end before interface FIFO interrupt enable 0 : disable 1 : enable
7	RO	0x0	reserved
6	W1C	0x0	BUS_ERR_EN axi master or ahb slave response error interrupt enable 0 : disable 1 : enable
5	RW	0x0	DFIFO_OF_EN DMA FIFO overflow interrupt enable 0 : disable 1 : enable
4	RW	0x0	IFIFO_OF_EN interface FIFO overflow interrupt enable 0 : disable 1 : enable
3	W1C	0x0	PIX_ERR_EN the pixel number of last line not equal to the set height interrupt enable 0 : disable 1 : enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	W1C	0x0	LINE_ERR_EN the line number of last frame not equal to the set height interrupt enable 0 : disable 1 : enable
1	W1C	0x0	LINE_END_EN line end interrupt enable 0 : disable 1 : enable
0	W1C	0x0	DMA_FRAME_END_EN dma frame end interrupt enable 0 : disable 1 : enable

**VIP\_INTSTAT**

Address: Operational Base + offset (0x00008)

VIP interrupt status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9	RW	0x0	PST_INF_FRAME_END frame end after interface FIFO interrupt 0 : no interrupt 1 : interrupt
8	RW	0x0	PRE_INF_FRAME_END frame end before interface FIFO interrupt 0 : no interrupt 1 : interrupt
7	RO	0x0	reserved
6	W1C	0x0	BUS_ERR axi master or ahb slave response error interrupt 0 : no interrupt 1 : interrupt
5	RW	0x0	DFIFO_OF DMA FIFO overflow interrupt 0 : no interrupt 1 : interrupt
4	RW	0x0	IFIFO_OF interface FIFO overflow interrupt 0 : no interrupt 1 : interrupt

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	W1C	0x0	PIX_ERR the pixel number of last line not equal to the set height interrupt 0 : no interrupt 1 : interrupt
2	W1C	0x0	LINE_ERR the line number of last frame not equal to the set height interrupt 0 : no interrupt 1 : interrupt
1	W1C	0x0	LINE_END line end interrupt 0 : no interrupt 1 : interrupt
0	W1C	0x0	DMA_FRAME_END dma frame end interrupt 0 : no interrupt 1 : interrupt

**VIP\_FOR**

Address: Operational Base + offset (0x0000c)

VIP format

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19	RW	0x0	UV_STORE_ORDER UV storage order 0 : UVUV 1 : VUVU
18	RW	0x0	RAW_END raw data endian 0 : little end 1 : big end
17	RW	0x0	OUT_420_ORDER output 420 order 0 : UV in the even line 1 : UV in the odd line Note: The first line is even line(line 0).
16	RW	0x0	OUTPUT_420 output 420 or 422 0 : output is 422 1 : output is 420
15	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14:13	RW	0x0	MIPI_MODE mipi mode 00 : 32 bit bypass mode; 01 : rgb mode; 10 : yuv mode; 11 : reserve;
12:11	RW	0x0	RAW_WIDTH raw data width 00 : 8bit raw data; 01 : 10bit raw data; 10 : 12bit raw data; 11 : reserve;
10	RW	0x0	JPEG_MODE JPEG mode 0 : other mode 1 : mode1
9	RW	0x0	FIELD_ORDER ccir input order 0 : odd field first 1 : even field first
8	RW	0x0	IN_420_ORDER 420 input order 0 : UV in the even line 1 : UV in the odd line Note: The first line is even line(line 0).
7	RW	0x0	INPUT_420 input 420 or 422 0 : 422 1 : 420
6:5	RW	0x0	YUV_IN_ORDER YUV input order 00 : UYVY 01 : YVYU 10 : VYUY 11 : YUYV
4:2	RW	0x0	INPUT_MODE input mode 000 : YUV 010 : PAL 011 : NTSC 100 : RAW 101 : JPEG 110 : MIPI Other : invalid

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	HREF_POL href input polarity 0 : high active 1 : low active
0	RW	0x0	VSYNC_POL vsync input polarity 0 : low active 1 : high active

**VIP\_FRM0\_ADDR\_Y**

Address: Operational Base + offset (0x00014)

VIP frame0 y address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	FRM0_ADDR_Y frame0 y address

**VIP\_FRM0\_ADDR\_UV**

Address: Operational Base + offset (0x00018)

VIP frame0 uv address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	FRM0_ADDR_UV frame0 uv address

**VIP\_FRM1\_ADDR\_Y**

Address: Operational Base + offset (0x0001c)

VIP frame1 y address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	FRM1_ADDR_Y frame1 y address

**VIP\_FRM1\_ADDR\_UV**

Address: Operational Base + offset (0x00020)

VIP frame1 uv address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	FRM1_ADDR_UV frame1 uv address

**VIP\_VIR\_LINE\_WIDTH**

Address: Operational Base + offset (0x00024)

VIP virtual line width

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	RO	0x0	reserved
14:0	RW	0x0000	VIR_LINE_WIDTH virtual line width

**VIP\_SET\_SIZE**

Address: Operational Base + offset (0x00028)

VIP frame set size

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x01e0	SET_HEIGHT set height
15:13	RO	0x0	reserved
12:0	RW	0x02d0	SET_WIDTH set width

**VIP\_CROP**

Address: Operational Base + offset (0x00044)

VIP crop start point

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x0000	START_Y start y point
15:13	RO	0x0	reserved
12:0	RW	0x0000	START_X start x point

**VIP\_SCL\_CTRL**

Address: Operational Base + offset (0x00048)

VIP scale control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6	RW	0x0	MIPI_32B_BP miipi 32 bit bypass 0 : no bypass 1 : bypass
5	RW	0x0	RAW_16B_BP raw 16 bit bypass 0 : no bypass 1 : bypass

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RW	0x0	YUV_16B_BP YUV 16 bit bypass 0 : no bypass 1 : bypass
3:0	RO	0x0	reserved

**VIP\_FIFO\_ENTRY**

Address: Operational Base + offset (0x00054)

VIP FIFO entry

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:18	RO	0x0	reserved
17:9	RW	0x000	UV_FIFO_ENTRY valid UV double word in FIFO write 0 clear
8:0	RO	0x000	Y_FIFO_ENTRY valid Y double word in FIFO write 0 clear

**VIP\_FRAME\_STATUS**

Address: Operational Base + offset (0x00060)

VIP frame status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0000	FRAME_NUM complete frame number write 0 to clear
15:2	RO	0x0	reserved
1	RO	0x0	F1_STS frame 0 status 0 : frame 1 not ready 1 : frame 1 ready write 0 clear
0	RO	0x0	F0_STS frame 0 status 0 : frame 0 not ready 1 : frame 0 ready write 0 clear

**VIP\_CUR\_DST**

Address: Operational Base + offset (0x00064)

VIP current destination address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	CUR_DST current destination address maybe not the current, because the clock synchronization.

**VIP\_LAST\_LINE**

Address: Operational Base + offset (0x00068)

VIP last frame line number

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12:0	RO	0x0000	LAST_LINE_NUM line number of last frame

**VIP\_LAST\_PIX**

Address: Operational Base + offset (0x0006c)

VIP last line pixel number

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28:16	RW	0x0000	LAST_UV_NUM y number of last line
15:13	RO	0x0	reserved
12:0	RO	0x0000	LAST_Y_NUM y number of last line

**30.5 Interface description**

<b>Module Pin</b>	<b>Direction</b>	<b>Pad Name</b>	<b>IOMUX Setting</b>
vip_clkout	O	IO_CIFclkin_HOSTTwkack_GPSclk_HSADCclkout_DV_Pgpio2b2	GRF_GPIO2B_MUX[5:4]==2'b1
vip_clkin	I	IO_CIFclkout_HOSTTwkreq_HSADCTSfail_DVPgpio2b3	GRF_GPIO2B_MUX[7:6]==2'b1
vip_href	I	IO_CIFhref_HOSTdin7_HSADCTSvalid_DVPgpio2b1	GRF_GPIO2B_MUX[3:2]==2'b1
vip_vsync	I	IO_CIFvsync_HOSTdin6_HSADCTSsync_DVPgpio2b0	GRF_GPIO2B_MUX[1:0]==2'b1
vip_data0	I	IO_CIFdata0_DVPgpio2b4	GRF_GPIO2B_MUX[9:8]==2'b1
vip_data1	I	IO_CIFdata1_DVPgpio2b5	GRF_GPIO2B_MUX[11:10]==2'b1
vip_data2	I	IO_CIFdata2_HOSTdin0_HSADCdata0_DVPgpio2a0	GRF_GPIO2A_MUX[1:0]==2'b1
vip_data3	I	IO_CIFdata3_HOSTdin1_HSADCdata1_DVPgpio2a1	GRF_GPIO2A_MUX[3:2]==2'b1
vip_data4	I	IO_CIFdata4_HOSTdin2_HSADCdata2_DVPgpio2a2	GRF_GPIO2A_MUX[5:4]==2'b1
vip_data5	I	IO_CIFdata5_HOSTdin3_HSADCdata3_DVPgpio2a3	GRF_GPIO2A_MUX[7:6]==2'b1
vip_data6	I	IO_CIFdata6_HOSTckinp_	GRF_GPIO2A_MUX[9:8]

		HSADCdata4_DVPgpio2a4	==2'b1
vip_data7	I	IO_CIFdata7_HOSTckinn_HSADCdata5_DVPgpio2a5	GRF_GPIO2A_MUX[11:10]==2'b1
vip_data8	I	IO_CIFdata8_HOSTdin4_HSADCdata6_DVPgpio2a6	GRF_GPIO2A_MUX[13:12]==2'b1
vip_data9	I	IO_CIFdata9_HOSTdin5_HSADCdata7_DVPgpio2a7	GRF_GPIO2A_MUX[15:14]==2'b1
vip_data10	I	IO_CIFdata10_DVPgpio2b6	GRF_GPIO2B_MUX[13:12]==2'b1
vip_data11	I	IO_CIFdata11_DVPgpio2b7	GRF_GPIO2B_MUX[15:14]==2'b1

## 30.6 Application Notes

The biggest configuration requirement of all operations is the CAP\_EN bit must be set after all the mode selection is ready. The configuration order of the input/output data format, YUV order, the address, frame size/width, AXI burst length and other options do not need to care.

There are many debug registers to make it easy to read the internal operation information of VIP. The valid pixel number of scale result in FIFO can be known by read VIP\_SCL\_VALID\_NUM. The line number of last frame and the pixel number of last line can be also known by read the VIP\_LAST\_LINE and VIP\_LAST\_PIX.

## Chapter 31 Image Signal Processing (ISP)

### 31.1 Overview

The Image Signal Processing (ISP) represents a complete video and still picture input unit. It contains image processing, scaling, and compression functions. The integrated image processing unit supports simple CMOS sensors delivering RGB Bayer pattern without any integrated image processing and also image sensors with integrated YCbCr processing.

Scaling is used for downsizing the sensor data for either displaying them on the LCD, or for generating data stream for MPEG-4 compression. In general, YCbCr 4:2:2 JPEG compressed images should use the full sensor resolution, but they can also be down-scaled to a lower resolution for smaller JPEG files. Scaling also can be used for digital zoom effects, because the scalers are capable of up-scaling as well.

An image effects block is present which can create images with sepia, black&white, color selection, negative, emboss and sketch effects.

The camera interface provides SMIA and/or MIPI support, so that ISP can be connected to PHY devices or IP blocks directly.

All data is transmitted via the memory interface to a BVCI/AXI bus system using a bus master interface.

Programming is done by register read/write transactions using an PVCI slave interface.

ISP supports the following features:

- Generic Sensor Interface with programmable polarity for synchronization signals
- ITU-R BT 601/656 compliant video interface supporting YCbCr or RGB Bayer data
- 12 bit camera interface
- 12 bit resolution per color component internally
- YCbCr 4:2:2 processing
- Flash light control
- Mechanical shutter support
- Hardware JPEG encoder incl. JFIF1.02 stream generator and programmable quantization and Huffman tables
- Windowing and frame synchronization
- Frame skip support for video (e.g. MPEG-4) encoding
- Macro block line, frame end, capture error, data loss interrupts and sync. (h\_start, v\_start) interrupts
- Luminance/chrominance and chrominance blue/red swapping for YUV input signals
- Continuous resize support
- Buffer in system memory organized as ring-buffer
- Buffer overflow protection for raw data and JPEG files
- Asynchronous reset input, software reset for the entire IP and separate software resets for all sub-modules
- Interconnect test support
- Semi planar storage format
- Color processing (contrast, saturation, brightness, hue, offset, range)
- Power management by software controlled clock disabling of currently not needed sub-modules
- Display-ready RGB output in self-picture path (RGB888, RGB666 and RGB565)
- Rotation unit in self-picture path (90°, 180°, 270° and h/v flipping) for RGB output
- Read port provided to read back a picture from system memory
- Simultaneous picture read back, resizing and storing through self path while main path captures the camera picture

- Black level compensation
- Four channel Lens shade correction (Vignetting)
- Auto focus measurement
- White balancing and black level measurement
- Auto exposure support by brightness measurement in 5x5 sub windows
- Defect pixel cluster correction unit (DPCC) supports on the fly and table based pixel correction
- De-noising pre filter (DPF)
- Enhanced color interpolation (RGB Bayer demosaicing)
- Chromatic aberration correction
- Combined edge sensitive Sharpening / Blurring filter (Noise filter)
- Color correction matrix (cross talk matrix)
- Global Tone Mapping with wide dynamic range unit (WDR)
- Image Stabilization support and Video Stabilization Measurement
- Flexible Histogram calculation
- Digital image effects (Emboss, Sketch, Sepia, B/W (Grayscale), Color Selection, Negative image, sharpening)
- Solarize effect through gamma correction
- AXI 64 bit interface 32Bit Address range (two DMA-write ports and one DMA-read port)
- Up to 16 Beat Bursts depending on configured FIFO size
- 32 bit AHB programming interface
- Maximum input resolution of 4416x3312 pixels
- Main scaler with pixel-accurate up- and down-scaling to any resolution between 4416x3312 and 32x16 pixel in processing mode
- Self scaler with pixel-accurate up- and down-scaling to any resolution between 1920x1080 and 32x16 pixel in processing mode
- Support of semiplanar NV21 color storage format
- Support of independent image cropping on main and self path

## 31.2 Block Diagram

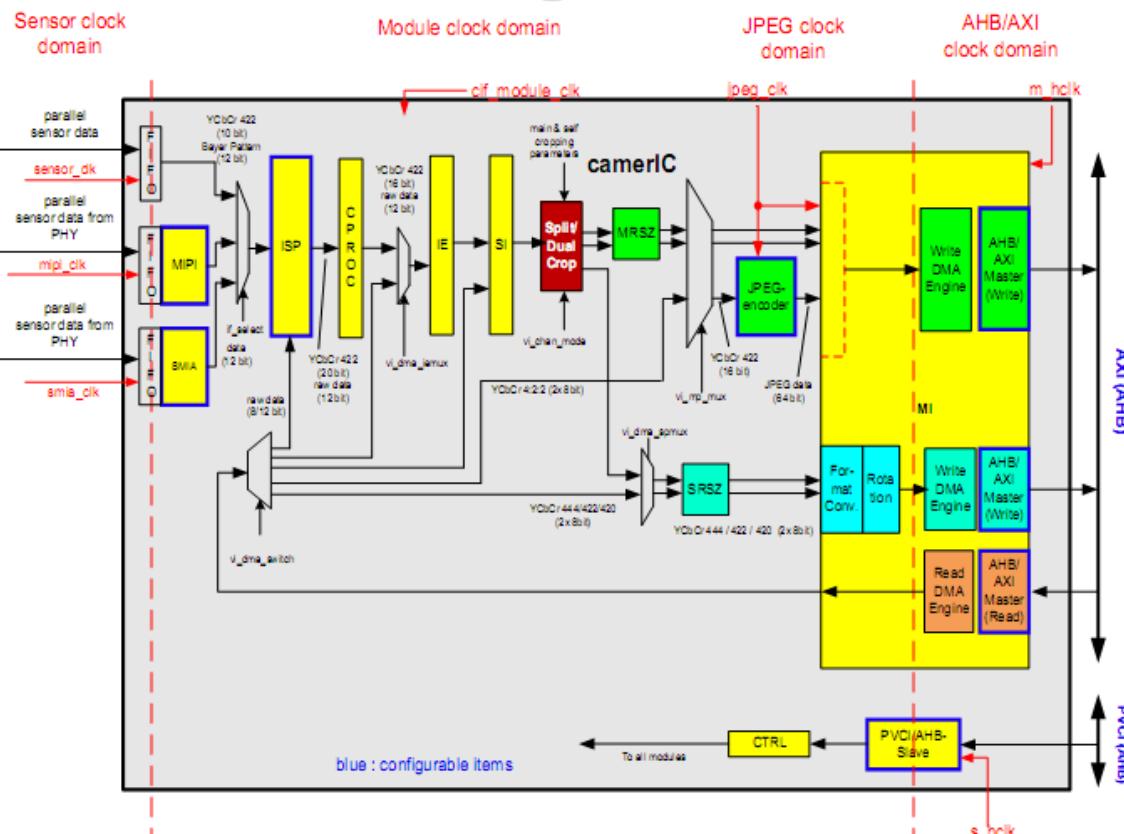


Fig. 31-1 ISP Block Diagram

ISP comprises with:

- MIPI serial camera interface
- Image Signal Processing (ISP)
- Color Processing (CPROC)
- Image Effects (IE)
- Superimpose (SI)
- Luminance /Chrominance Splitter (Y/C Split)
- Main and Self Crop (Dual Crop)
- Main Resize (MRSZ)
- JPEG Encoder
- MPMUX for selection of main path data flow
- SPMUX for selection of self-scaler input data
- Self Resize (SRSZ)
- Memory Interface (MI) including YCbCr to RGB conversion for self-picture and image rotation
- Control Unit

## 31.3 Function Description

### 31.3.1 MIPI

The MIPI interface is the second optional serial camera interface of ISP Controller. The interface is implemented according to the CSI2 specification defined by the MIPI Alliance, connecting a PPI data interface to a MIPI\_D-PHY physical layer with a 12bits ALOMICs interface.

#### Features and Standard Compliance

Compliant to MIPI Alliance Standard for Camera Serial Interface 2 (CSI2)

- PPI Interface according to D-PHY specification, Annex A
- Supports up to 4 data lanes (extendable by request)
- Number of lanes is programmable and hardware configurable
- Provides lane merging, error detection and correction, virtual channel detection, programmable data extraction and embedded data separation
- Supported data types are:
  - Generic 8bit data
  - Non-legacy YUV 4:2:0 8bit / 10bit with cosited chroma sampling
  - Non-legacy YUV 4:2:0 8bit / 10bit with non-cosited chroma sampling
  - Legacy YUV 4:2:0 8bit
  - YUV 4:2:2 8bit / 10bit
  - RGB 444 / 555 / 565 / 666 / 888 image data
  - RAW 6-bit / 7-bit / 8-bit / 10-bit / 12-bit image data
  - User-defined 8-bit data
- PVCi similar output interface
- PVCi control interface

RAW 14-bit image data types are not supported. The MIPI D-PHY-Layer features Escape Modus and Low Power Data Transfer are not needed for the Protocol Receiver. According to the CSI2 specification only unidirectional high-speed data transfer is mandatory for the camera interface.

### 31.3.2 ISP Block

The ISP block includes the interface to the attached parallel sensor device and the MIPI interfaces. It accepts either ITU-R BT 601 YCbCr, well as raw Bayer data or ITU-R BT 656 YCbCr data. The order of Y and Cx as well as Cb and Cr is programmable. It contains the interpolation filter for the raw Bayer to plain RGB conversion.

An input acquisition window is programmable supporting detection of smaller than programmed input images. An error IRQ is generated in case of input error conditions.

The ISP allows programming of a continuous image sampling mode, or a mode where the number of images to be sampled subsequently can be programmed (1...1023).

Additionally it incorporates image quality improvements (gamma correction, black level subtraction, white balancing, etc.).

The ISP block supports also auto exposure capabilities by providing image color statistics information to the system processor for correct sensor device programming.

The input part of the ISP block is fully programmable in terms of signal polarities, active video data positions, and luminance/chrominance order.

The ISP block always delivers YCbCr 4:2:2 data at its output port. Data provided by the RGB channel are downsampled to YCbCr 4:2:2, supporting both co-sited and non co-sited calculation. Luminance and chrominance data are provided in parallel using line and frame end signals. In raw data mode (unprocessed data) these data are transferred via the Y port.

Handshaking is used for data qualification. As the sensor device cannot be stopped delivering data the backward handshake (acknowledge) is only used for pixel drop detection which is signaled using an IRQ. To prevent pixel dropping a latency FIFO is implemented. The FIFO depth can be option of customization.

The ISP block also contains its own programming registers to be accessed by a 32 bit PVCI compliant interface supporting single transfers only.

The incoming hsync and vsync control signals from the camera are connected to interrupt logic. It is possible to trigger on these signals for an event triggered configuration during processing.

## **DPCC**

The ISP core is designed to operate with high-end, mid-range and low-end image sensors, which mainly differ in the number of pixels and module cost. While high-end sensors aim at a high number of pixels with sufficiently large pixel area, low-cost modules often have a large number of defects. Additionally, for low cost sensors in the manufacturing process no time is available for determining the defect pixels locations.

Another problem is that hot pixels get visible at long integration times and can get a high density up to 5%. This means 250000 defect pixels for a 5 Megapixel sensor.

An improved algorithm has been developed, named Defect Pixel Cluster Correction (DPCC). This is an on-the-fly detection and replacement processing as well as a table based replacement method

### **Defect Pixel Detection**

For each pixel threshold values are calculated by several methods, using the correlation of neighbor pixel of the same color (red, green or blue) with exception of the peak gradient estimation for red and blue that also uses the green pixel values in the 5x5 neighborhood of the raw Bayer image. These methods use statistical properties and linear prediction to determine if a pixel needs to be marked as defect. A 3x3 sorting algorithm with rank estimation including the calculation of median values of some pixel groups is a central unit. Output of the detection unit is a marker signal for the following correction stage to indicate, if the current pixel is defect. The detection can be controlled by programmable threshold values, factors and options which methods should be used. The correct setting of the thresholds is important for a good separation between defects and keeping high resolution and detail features in the image.

### **Defect Pixel Table control**

A defect pixel table is implemented as SRAM of user defined size containing entries with defect

pixel coordinates. The table control generates a replace flag independently from the on-the-fly detection, so that table correction and on-the-fly correction can be operated in parallel.

## Defect Pixel Replacement

Basic algorithm is a switching median filter, which performs sorting and rank ordering. The replacement unit takes the information of the Defect Pixel Detection as input. It uses a statistical sorting filter (median filter) separately for each color to determine the nearest neighbor value for replacement. The filter size is 4 (upper, lower, left and right neighbor for red/blue or diagonal neighbors for green) optionally 5, including the center pixel. The condition for successfull replacement is that NOT more than 4 of 9 pixels are defect. So correction of single and small cluster defects of 1x3 (3x1) and 2x2 pixels is possible (please note these clusters belong to a single color in the bayer pattern).

## 9. Auto Focus Measurement

An auto-focus measurement block is implemented to support auto-focus control. A substantial part of auto-focus control will be done software supported: The search algorithm which looks for maximum sharpness in the image is implemented using software and the movement of the lens is controlled by software. The auto-focus module which is implemented using hardware delivers measurement values of image sharpness via a register interface.

The module measures the sharpness in 3 windows of selectable size via register settings. The auto-focus measurement block uses the line buffer of the emosaicing block. The data in the buffer are stored in Bayer format and read 3-line-wise (top, middle, bottom).

## 10. Filter (Noise Reduction, Sharpness, Blurring)

As mentioned above, high-end, mid-range and low-end image sensors mainly differ in the number of pixels and module cost. While high end sensors aim at a high number of pixels with sufficiently large pixel area, low-cost modules have small pixels and lens with small diameter. This combination of small lens diameter and small pixels results in higher pixel noise at a given level of illumination than in high end sensors.

To improve the visual image quality, noise and artifacts of the Bayer Matrix should be eliminated and sharpness should be increased. These are requirements which cannot be easily combined.

Noise and the artifacts of the Bayer pattern must be eliminated by averaging or blurring filters. Sharpness could be improved by a high pass or Laplace filter.

Texture detection allows detecting planes or edges, high or low density of details.

With this texture information an adaptive filter is controlled which reduces noise in planes and improves sharpness in detailed regions. If there are less details below or near the noise level two-dimensional blurring is applied. With a higher detail level, the blurring is done along detected edges or lines. If improvement of sharpness is required, the sharpness will be enhanced orthogonally to the direction of the blurring operation.

For example if a horizontal edge is detected, blurring will be done in horizontal direction and sharpness will be improved in vertical direction. In regions with highest contrast and details the filter can be bypassed or if sharpness improvement is required, a two dimensional sharpness filter is realized.

Additionally if no noise reduction is required a fixed blurring or sharpness filter can be selected. The sharpness can be improved by a fixed sharpness filter or edge depending as described beyond.

The noise reduction level can be adapted by the filter coefficients which determine the weightiness of averaging, and by the threshold values which are compared with the texture detection results.

For an optimal noise reduction it is important to know the effective noise level. If the noise

level is too high, the noise cannot be eliminated at the best possible rate. If the noise level is low and the noise reduction level is too high, too much details of the image will be lost unnecessarily.

If the light conditions are known, the noise level can be estimated using a table from the exposure settings of the image sensor. If the noise level must be calculated e.g. from the variance of the image RGB data, it is difficult to decide if the variance is caused by the noise or by image patterns.

In addition the hardware effort for separate filters is high because of the necessary line buffers. Therefore the filter algorithm is combined with demosaicing, so that the same line buffer can be used for demosaicing, noise filtering and blurring/sharpening.

The combined demosaicing filter module is designed to operate with linear RGB white balanced Bayer Data. So for best results of interpolation for demosaicing and of texture detection the RGB data should be optimal white balanced.

## **11. Video Stabilization**

The ISP Controller Video Stabilization consists of the following components

- Video Stabilization Measurement
- Video Stabilization Software
- ISP Image Stabilizer

The Video Stabilization Measurement is done in hardware. It calculates per image the horizontal and vertical displacement vector for global motion in comparison to the previous image.

The Video Stabilization Software reads these measurements values, applies some smoothing algorithms and provides the respective cropping window to the ISP Image Stabilization.

The ISP Image Stabilization crops the image with respect to the given cropping window.

The cropping of an image has to be done after the video stabilization measurement for the image has been completed. This involves the usage of at least 1 frame buffer in system memory.

The Image Stabilizer unit delivers YUV data (either unmodified YUV data received as ITU-R BT.601 or ITU-R BT.656 data or color-processed image data) to the following ISP Controller environment.

The Video Stabilization process is currently limited to input line sizes of 2048 pixels and output line sizes of 1280 pixels.

## **12. Mechanical Shutter Control**

With increasing sensor resolution the expectation of high quality pictures also rises. Therefore the exposure control must be decoupled from the frame rate and data read-out. This can be achieved by a mechanical shutter. The mechanical shutter control supports shutter speeds from 1/4000 sec to 10.7 sec by 48-100MHz.

## **13. Flash Light Control**

The sensor interface supports triggering of a LED or tube flash light. The flash light output and the prelight output can be used to control a flash light device. Both the flash light and the prelight are activated by a trigger event. This event may either be a positive or negative edge from the camera or a positive edge from any other trigger source at the input port 'vds\_vsync'.

Signal polarity, flash delay time and flash light time are determined by programming respective configuration register.

## **14. Histogram Calculation**

A histogram function is implemented which counts the number of pixels with the same value. In general this histogram is a graphical representation of the pattern of variation that exists in the intensity values of the color or luminance planes.

Usually it is displayed by vertical bars drawn to indicate frequency levels of data collected within specific ranges. This measurement block can be used for different purposes. The most obvious application is an informative display for the end user. It is used for improving the exposure control, which is done by a software control loop.

Histogram Calculation is done independently for 5x5 windows which is important for advanced and fast auto exposure algorithms with complex scene detection.

## **15. Lens Shading Correction**

The lens shading correction deals with the problems of vignetting and lens shading. It is done during input data processing: If the lens shading correction is enabled, each pixel is processed and corrected according to the stored settings. The lens shading correction is done by multiplying each input pixel with its respective correction value.

Only the correction factors at predefined sector corners as well as the sector positions are stored. The pixel position specific correction values are calculated using bilinear interpolation.

The correction factors at the sector corners are calculated during a calibration process which uses one or more reference frames which have to be captured under dedicated light conditions and at a dedicated position of the sensor. The captured frames are evaluated by software and the calculated parameters for lens shading correction are stored in multiple illumination specific tables e.g. in external memory or on a flash device. The software controls the lens shading process by loading or updating the correct tables into the hardware module.

It is also possible to use different lens shading correction parameters for different environment conditions, e.g. lightness, light direction or sensor position.

## **16. Wide Dynamic Range (WDR)**

ISP contains a global tone and color mapping unit for Wide Dynamic Range (WDR) compression. Compared with a standard camera, the dynamic range of input intensities appears to be widened, since more structure becomes visible out of the dark and bright image regions.

The dynamic range of real-word scenes is much higher than the available dynamic range of low cost CMOS image sensors. The image sensor thus captures a small range of the real word's scene radiance and maps it to the available output range of the sensor. Radiance levels above or below the sensor's value range are clipped to black or white in the sensor.

The auto-exposure control (AEC) controls which portion of the scene radiance is mapped to the sensor value range. The AEC uses a model-driven scene evaluation to determine the best exposure value.

Nevertheless, there is a chance that portions in the scene are mapped to the dark grey tones or near white tones. This is the case especially in high contrast scenes, when there is anyhow not the chance to perfectly reproduce the full scene radiance range.

Global tone mapping can be used to reduce this effect. It aims at shifting textures in dark grey or near white tones into the mid tone range and thus allows to optimize the perceptual reproduction of the scene. Compared with a standard camera, the dynamic range of input intensities appears to be widened, since more structure becomes visible out of the dark and bright image regions.

This step is being performed directly before the Gamma-Correction. Basically by applying a scene dependent tone curve the required intensity shift is being performed.

During this step the following constraints have to be considered:

- Shifting of textures from dark grey into mid tones increases noise. Thus this step should only be performed for images with a sufficiently low noise level in dark grey tones.
- Changing the intensity level of a pixel also effects the color saturation. In order to avoid color clipping a correction of the color saturation is to be performed. This ensures that after tone mapping the colors have the same hue as before.

The tone and color clipping correction unit for wide dynamic range applications (WDR) for MARVIN performs scene dependent correction such as brightening of dark texture tones. Suitably, the MARVIN denoising pre-filter is being used for edge-preserving noise reduction especially in dark textures. Additionally, color clipping compensation is being performed with tone mapping.

### **31.3.3 Color Processing**

The Color Processing block is responsible for color processing functions, i.e. hue, contrast, brightness, and saturation adjustment. It operates at YCbCr 4:2:2 data.

### **31.3.4 Image Effects (IE)**

The Image Effects block modifies an image by pixel modifications. A set of different modifications can be applied: grayscale-, sepia-, color selection-, negative-, emboss-, sketch and sharpening effects

In addition a solarize effect can be created by using the gamma block of ISP Controller (for more information an application note is available).

The Image Effects module gets YCbCr 4:2:2 data via a 16 bit ([15:8]: Y, [7:0]: Cb/Cr) data interface.

#### **IE Feature**

- Data input- and output handshake interface
- Supports YCbCr 4:2:2 format
- 3x3 Laplace Filter (for picture edge extraction used for emboss, sketch and sharpening effects)
- ITU-R BT.601 compliant YCbCr to RGB conversion (used for the color selection effect to compare red green blue components of a pixel with defined threshold values)

### **31.3.5 Super Impose (SI)**

The Super Impose module overlays an image with a bitmap from the main memory

- Color of the transparent area in superimpose bitmap is configurable. So the camera picture interfuses through the transparent area (A).
- Furthermore the Superimpose block is able to position a bitmap with the appropriate coordinates over the camera image range (B).

The Super Impose module gets picture data in YCbCr 4:2:2 formats via a 16 bit data interface from the Image Effects module. The Memory Interface module delivers the Y, Cb and Cr pixel components of the superimpose bitmap through three independent handshake data interfaces. Within the common area of the two pictures, output pixel data is determined by the bitmap from main memory or by the image from the Image Effects module.

### **31.3.6 Luminance/Chrominance Splitter (YC\_Split)**

The Luminance / Chrominance Splitter is responsible for providing component separated YCbCr 4:2:2 pixel data for further processing. Therefore it has to split the data path for chrominance and luminance values as well as for main and self-picture path.

### 31.3.7 Resize (Main and Self Scaler)

The Resize module is instantiated twice in ISP Controller: One instance for the main path (main resize, MRSZ) and one for the self path (self resize, SRSZ). The Resize Modules get pictures in YCbCr 4:2:2 format, and scale them by an arbitrary factor up or down to a new format. See following Figure for a module overview:

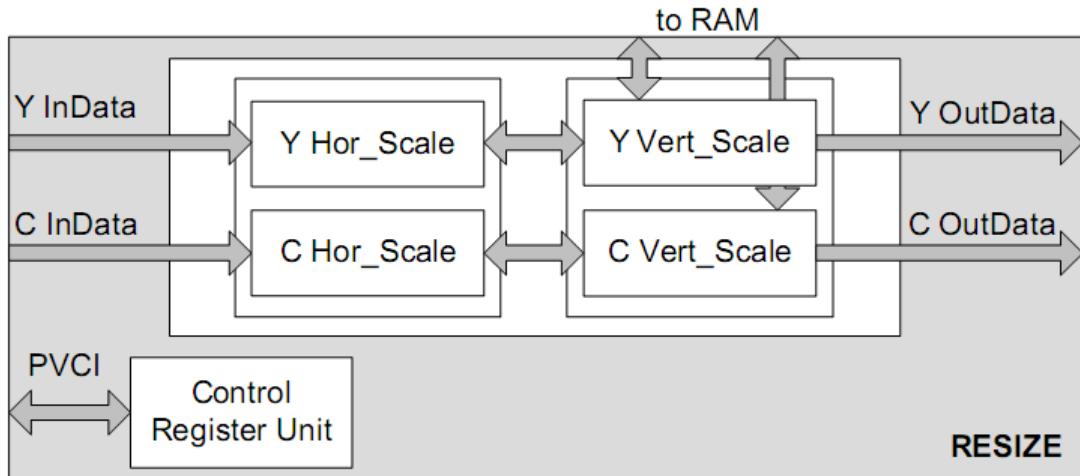


Fig. 31-2 Block Diagram of the Resize Module

#### Feature and Standard Compliance

PVCI compliant control port

- Data input and output handshake interfaces
- Supports 4:2:2 YCbYCr input format
- Different output formats (4:2:2, 4:2:0, 4:1:1, 4:1:0) possible by choosing from different scaling factors for luminance and chrominance components
- Support of cosited and non-cosited output formats via programmable phase offsets
- Discrete bypassing of each submodule is possible
- Output frame size for Main Picture Scaler (MRSZ) is up to 64 Mpixels.
- Output frame size for Self Picture Scaler (SRSZ) is independent from the Main Picture Scaler and is typically set to lower resolutions (e.g. Full HD). It can be object of customization.

The Resize module is configurable for horizontal and vertical up- or down-scaling.

Discrete values for the scaling factors of the luminance and the two chrominance components allow conversion between YUV4:2:2 and YUV4:2:0 color format and support of uneven line width.

Phase shift registers are provided to shift the output pixel positions with respect to the input pixel positions. This allows for e.g. format conversion between cosited and non-cosited color schemes.

In sensor mode the MRSZ block supports only down-scaling. This is because the sensor cannot be stopped from delivering data during one frame.

The Resize module is able to process luminance and chrominance data independently, i.e. there are separate pipelines for luminance and chrominance processing using dedicated scale factors and phase offsets. This allows format conversion to be done by the Resize block (YCbCr 4:2:2 to 4:2:0, 4:1:1, 4:1:0).

### 31.3.8 JPEG Encoder

The JPEG Encoder consists of the Raster-2-Block converter and the JPEG encoder core. It accepts YCbCr 4:2:2 data only.

The Raster-2-Block converter uses an optimized addressing algorithm for its block memory, so that a total of 8 line buffers are sufficient. The block memory operates in a limited way as a FIFO, so some delay in the processing behind the Raster-2-Block converter can be leveled.

The data input interface is a 16 bit YCbCr 4:2:2 interface. As the encoder core accepts 8 bit data only, the Raster-2-Block converter accepts 16 bit pixel data at maximum half the clock speed, i.e. after taking a 16 bit sample it de-asserts its acknowledge for one clock cycle.

The output interface is 64 bits wide. Both interfaces use handshaking. The programming interface is a 16 bit PVCI interface.

### 31.3.9 YC<sub>b</sub>C<sub>r</sub> to RGB Conversion

In the self-picture path a YCbCr to RGB conversion unit is present to provide display-ready image data for LCD displays. The image data is stored in one memory region of the system memory. These formats are supported: RGB 888 with 24 bit per pixel, RGB 666 with 18 bit per pixel and RGB 565 with 16 bit per pixel.

The conversion from YCbCr 4:2:2 to 4:4:4 mode is performed by storing the chrominance value of the previous pixel. Thus a set of Y, Cb and Cr values is available for every pixel to be processed. These values are used to generate an RGB value for the pixel by using the following formulas for 8 bit (0 – 255) RGB, assuming a nominal range of 16 to 235 for luminance and 16 to 240 for chrominance values:

$$R = 1,164*(Y - 16) + 1,596*(Cr - 128)$$

$$G = 1,164*(Y - 16) - 0,813*(Cr - 128) - 0,391*(Cb - 128)$$

$$B = 1,164*(Y - 16) + 2,018*(Cb - 128)$$

The result is forwarded to the MI as 24 bit word via the sp\_y\_data port, the chroma port (sp\_c\_data) is inactive in this case.

The module can be configured to support RGB 565 and RGB 666 by clipping the RGB 888 values. In case of RGB 565 only the lower 16 bits of the data port to the SP\_Y\_FIFO are used. In bypass mode (no RGB conversion) the conversion functionality is inactive and the incoming luminance and chrominance values are simply fed through to the MI. See the following Figure for the memory organization for the self picture path:

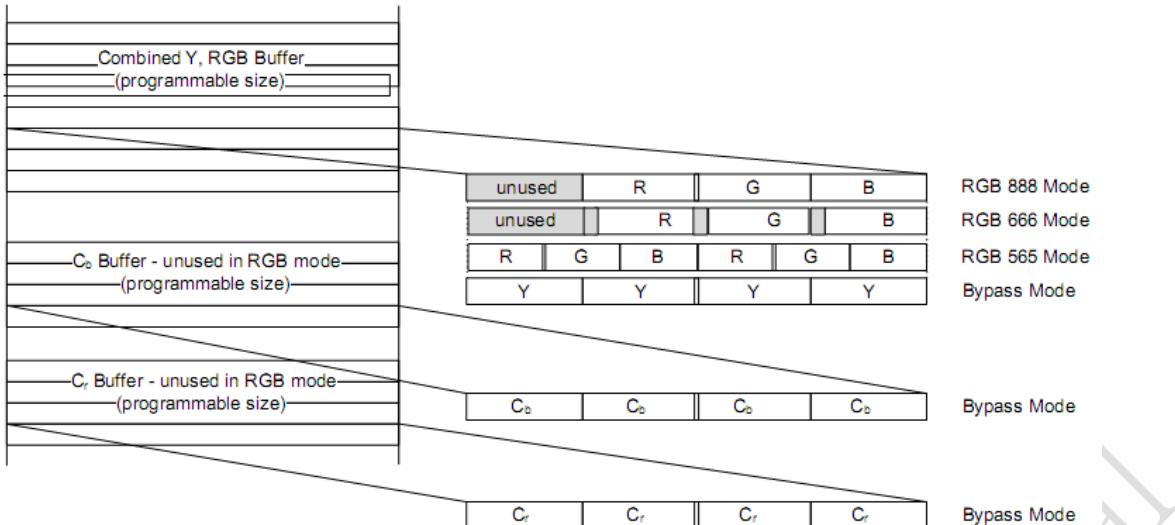


Fig. 31-3 Memory Organization for the Self Picture Path

### 31.3.10 Memory Interface (MI)

The Memory Interface block provides three data bus master ports to the system memory, two write ports and one read port.

#### Write Port

The Memory Interface is responsible for collecting the internal data streams and writing them into system memory. Therefore it is attached to data bus master wrappers to access (write) the system bus.

The following types of data streams are supported:

- Raw 8 or 12 bit data
- 64 bit JPEG data
- 2x 8 bit Y, Cb/Cr main image data
- 2x 8 bit Y, Cb/Cr self-picture data

The following modes of operation must be supported:

- Raw data only
- JPEG data only
- Main image data only
- Self-picture data only
- Parallel main image and self-picture data
- Self-picture and JPEG data
- Semi-planar mode

The image data has to be split into Y, Cb and Cr data to be separately written into system memory. So the Memory Interface (write port) consists of six FIFOs for the data.

The FIFOs are necessary for the component separated data streams of main and self-picture data path. Some of the FIFOs have to be re-used for the raw data and JPEG data stream. See Fig.25-4 for the definition of memory buffers and Fig.25-5 for the storage scheme in planar and semi-planar mode.

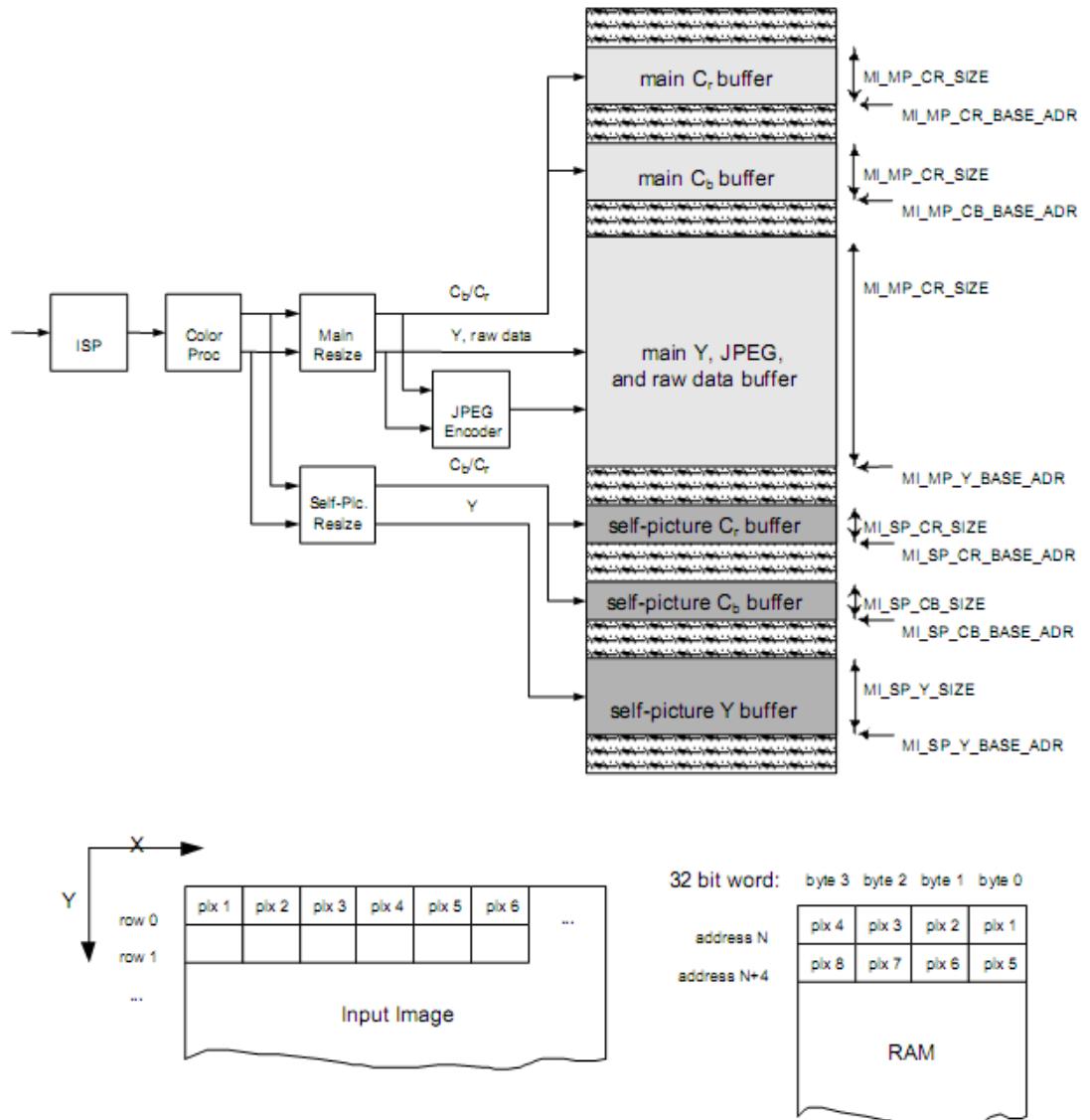


Fig. 31-4 Definition of Memory Buffers

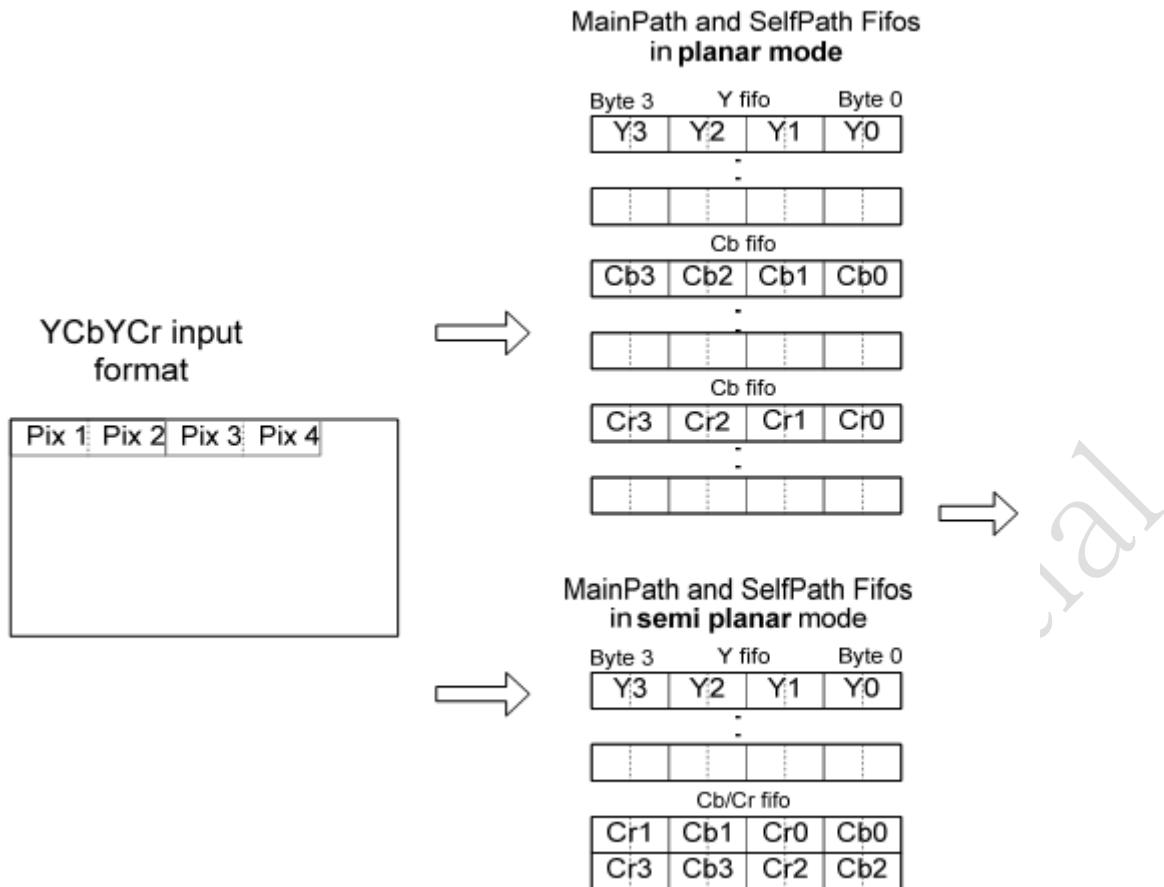


Fig. 31-5 Storage Scheme in Planar and Semi-planar Mode

### Read port (DMA)

The Memory Interface also supports reading back picture data from the system memory.

Therefore it is attached to a second data bus master wrapper to access (read) the system data bus.

Three independent read channels are provided to accommodate the three color components of a picture. The picture has to be stored component separated (planar) in system memory. If one or two components are not used they can be turned off by programming the respective component size to zero. So it's also possible to read back raw or JPEG data through one single channel.

The Memory Interface (read port) consists of three FIFOs, one FIFO per channel. Each FIFO features a PVCI interface at the output, so data can be easily halted by de-asserting the acknowledge line.

### 31.3.11 Control Unit

The Control Unit serves two purposes:

1. Interface o the local configuration register blocks of the other modules
2. Clock and reset control registers for ISP Controller core

The Control Unit has one 32 bit PVCI input interface and multiple 32 bit PVCI output interfaces. All transfers from the PVCI input interface are 32 bit wide.

Each block inside the ISP Controller IP core uses a dedicated clock signal that can be controlled by a programming register inside the Control Unit.

Existing software resets in the ISP Controller blocks can also be controlled by a programming register inside the Control Unit. An asynchronous reset for the processing clock domain has to be generated from the system reset. A soft reset for all registers in the IP core is provided. It works like an asynchronous reset.

All sub module processing clocks can be switched on and off. All sub module configuration clocks are enabled only when the dedicated address is active. All gated processing and configuration clocks are phase synchronous to the processing clock.

## 31.4 Register Description

The ISP uses a distributed configuration register scheme. So there is no central unit containing all programming registers, but all sub-modules contain their own programming registers. An address space is reserved for each sub-module inside the total ISP Controller address space.

### 31.4.1 Registers Summary

No.	Module	Description	Base Name	Offset Address
1	MX_main_control	ISP Main Control Registers	MRV_AFIM_BASE	0x0000
2	MX_image_effects	Image Effects	MRV_IMGEFF_BASE	0x0200
3	MX_superimpose	Superimpose	MRV_SI_BASE	0x0300
4	MX_isp	ISP main registers	MRV_ISP_BASE	0x0400
5	MX_isp_flash	FLASH_LIGHT registers	MRV_FLASH_BASE	0x0660
6	MX_isp_shutter	SHUTTER registers	MRV_SHUT_BASE	0x0680
7	MX_cproc	COLOR PROCESSING registers	MRV_CPROC_BASE	0x0800
8	MX_main_resize	MAIN RESIZE registers	MRV_MRSZ_BASE	0x0c00
9	MX_self_resize	SELF RESIZE registers	MRV_SRSZ_BASE	0x1000
10	MX_mi	Memory Interface registers	MRV_MI_BASE	0x1400
11	MX_jpeg	JPEG ENCODER registers	MRV_JPE_BASE	0x1800
12	MX_smia	SMIA Interface registers	MRV_SMIA_BASE	0x1a00
13	MX_mipi	MIPi Interface registers	MRV_MIPI_BASE	0x1c00
14	MX_isp_afm	ISP Auto Focus Measurement	MRV_AFIM_BASE	0x2000

No.	Module	Description	Base Name	Offset Address
15	MX_isp_lsc	ISP Lens Shade Correction	MRV_LSC_BASE	0x2200
16	MX_isp_is	ISP Image Stabilization	MRV_IS_BASE	0x2300
17	MX_isp_hist	ISP Histogram	MRV_HIST_BASE	0x2400
18	MX_isp_filter	ISP Filter	MRV_FILT_BASE	0x2500
19	MX_isp_cac	ISP Chromatic Aberration Correction	MRV_CAC_BASE	0x2600
20	MX_isp_exposure	ISP Auto Exposure Measurement	MRV_AE_BASE	0x2700
21	MX_isp_bls	ISP Black Level Subtraction	MRV_BLS_BASE	0x2800
22	MX_isp_dpf	ISP De-noising Pre-filter	MRV_DPF_BASE	0x2900
23	MX_isp_dpcc	ISP Defect Pixel Cluster Correction	MRV_DPCC_BASE	0x2900
24	MX_isp_wdr	ISP Wide Dynamic Range	MRV_WDR_BASE	0x2a00

Notes: **S**ize: **B**- Byte (8 bits) access, **H**W- Half WORD (16 bits) access, **W**-WORD (32 bits) access

### 31.4.2 Detail Register Description

The description of the ISP register reference **5.1 Register Interface of CamerIC2D\_user\_manual\_M14v2.pdf**.

## 31.5 Interface Description

Table 31-1 ISP Interface Description

Module Pin	IO	Pad Name	IOMUX Setting
isp_clkin	I	CIFclkin_GPIO2b3	GPIO2B_IOMUX[7:6]= 2'b01
isp_href	I	CIFhref_GPIO2b2	GPIO2B_IOMUX[5:4]= 2'b01
isp_vsync	I	CIFvsync_GPIO2b1	GPIO2B_IOMUX[3:2]= 2'b01
isp_data0	I	CIFdata0_I2C3scl_cam_HDMIddc_scl_GPIO1d4	GPIO1D_IOMUX[9:8]= 2'b01
isp_data1	I	CIFdata1_I2C3sda_cam_HDMIddc_sda_GPIO1d5	GPIO1D_IOMUX[11:10]= 2'b01
isp_data2	I	CIFdata2_HSADCdata0_BBdebug0_GPIO2a0	GPIO2A_IOMUX[1:0]= 2'b01

isp_data3	I	CIFdata3_HSADCdata1_BBdebug1_GPIO2a1	GPIO2A_IOMUX[3:2]= 2'b01
isp_data4	I	CIFdata4_HSADCdata2_BBdebug2_GPIO2a2	GPIO2A_IOMUX[5:4]= 2'b01
isp_data5	I	CIFdata5_HSADCdata3_BBdebug3_GPIO2a3	GPIO2A_IOMUX[7:6]= 2'b01
isp_data6	I	CIFdata6_HSADCdata4_BBdebug4_GPIO2a4	GPIO2A_IOMUX[9:8]= 2'b01
isp_data7	I	CIFdata7_HSADCdata5_BBdebug5_GPIO2a5	GPIO2A_IOMUX[11:10]= 2'b01
isp_data8	I	CIFdata8_HSADCdata6_BBdebug6_GPIO2a6	GPIO2A_IOMUX[13:12]= 2'b01
isp_data9	I	CIFdata9_HSADCdata7_BBdebug7_GPIO2a7	GPIO2A_IOMUX[15:14]= 2'b01
isp_data01	I	CIFdata01_GPIO1d6	GPIO1D_IOMUX[13:12]= 2'b01
isp_data11	I	CIFdata11_GPIO1d7	GPIO1D_IOMUX[15:14]= 2'b01

Notes: 1. I=*input*, O=*output*, I/O=*input/output*, bidirectional

## 31.6 Application Notes

The description of the Application Notes reference **5.1 Register Interface of CamerIC2D\_user\_manual\_M14v2.pdf**.

## Chapter 32 HDMI TX

### 32.1 Overview

HDMI TX is fully compliant with HDMI 1.4a and 2.0a specification. It offers a simple implementation for consumer electronics like DVD/player/recorder and camcorder. HDMI TX consists of one HDMI transmitter controller and one HDMI transmitter PHY.

It supports following features:

- Video formats:
  - All CEA-861-E video formats up to 1080p at 60 Hz and 720p/1080i at 120 Hz
  - Optional HDMI 1.4b video formats
  - HDMI 2.0 video formats, All CEA-861-F video formats
- Colorimetry, 24/30-bit RGB 4:4:4
- Pixel clock from 13.5 MHz up to 600 MHz
- Up to 192 kHz IEC60958 audio sampling rate
- Flexible synchronous enable per clock domain to set functional power down modes
- AMBA APB 3.0 register access
- I2C DDC, EDID block read mode
- SCDC I2C DDC access
- TMDS Scrambler to enable support for 2160p@60Hz with RGB 4:4:4
- Integrated CEC hardware engine

### 32.2 Block Diagram

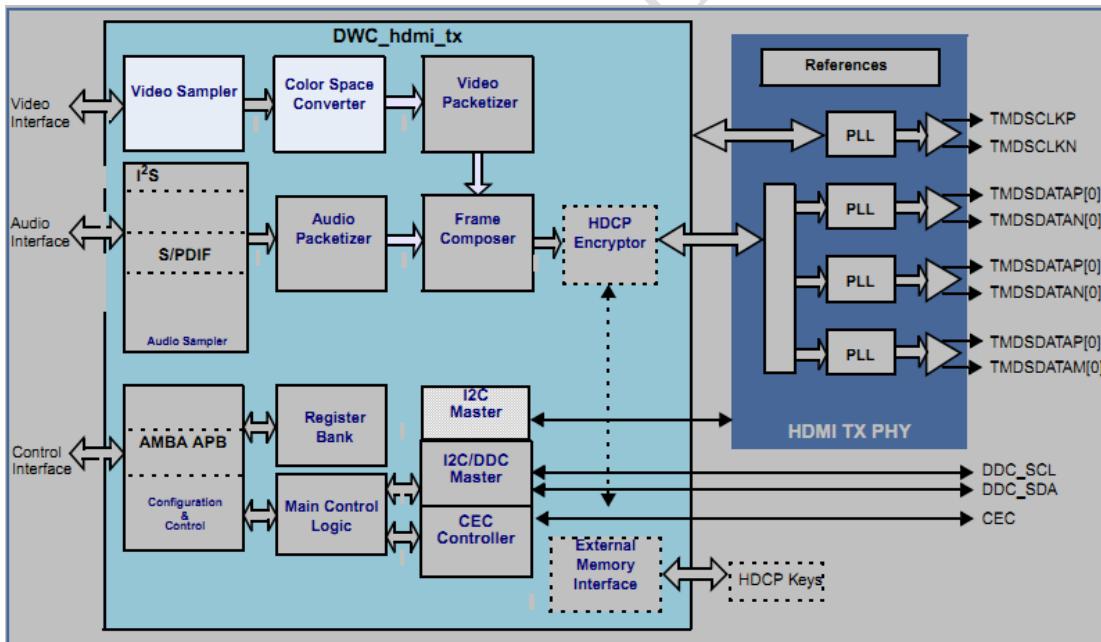


Fig. 32-1 HDMI TX Block Diagram

### 32.3 Function Description

#### 32.3.1 Video Data Processing

The video processing contain video format timings, pixel encodings(RGB to YCbCr, or YCbCr to RGB), colorimetry and corresponding requirements. This function is implemented by some functional blocks, Video Capture block, Color Space Conversion block, and Deep Color block.

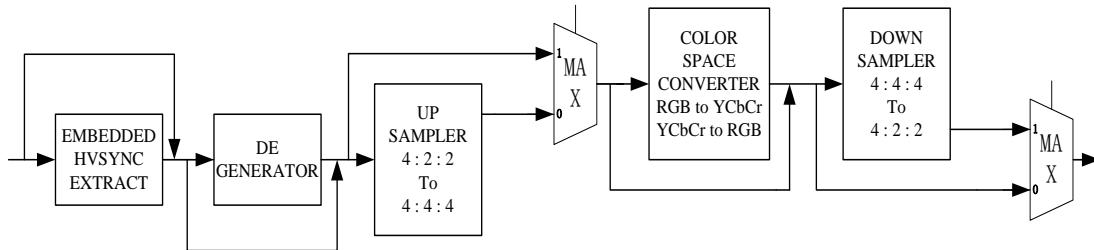


Fig. 32-2 HDMI Video Data Processing

The input video pixels can be encoded in either RGB, YCBCR 4:4:4 or YCBCR 4:2:2 formats by Color Space Conversion block.

The input Video data can have a pixel size of 24, 30bits. The deep color block is used to deal with different pixel size. Video at the default 24-bit color depth is carried at a TMDS clock rate equal to the pixel clock rate. Higher color depths are carried using a correspondingly higher TMDS clock rate. HDMI Transmitter support video formats with TMDS rates below 25MHz (e.g. 13.5MHz for 480i/NTSC) that can be transmitted using a pixel-repetition scheme by setting relative registers.

The following interface timing diagram outlines the Video interface signal format. 24 bit data (we also support 36 bit data for deep color) in RGB can be captured by the rising edge of VCLK with 1ns setup time and 1ns hold time requirements. Control signals such DE and VSync/HSync/FSync going with the same timing relationship.

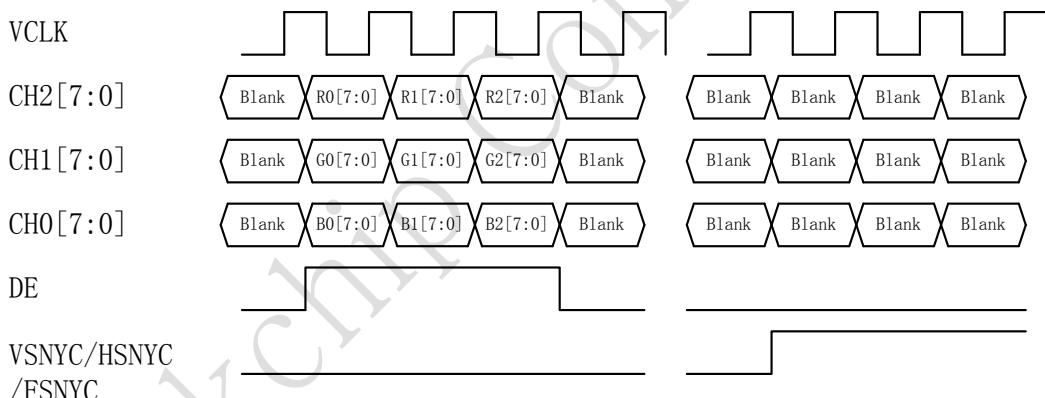


Fig. 32-3 HDMI Video Processing Timing

## 1. Video Data Capture Logic

HDMI TX support input video data related format table is listed below.

Table 32-1 HDMI Supported Input Video Formats

Color Space	Pixel Encoding	Sync	Channel Width	Pin Nums
RGB	4:4:4	Separate	8	24
RGB	4:4:4	Separate	10	30
RGB	4:4:4	Separate	12	36
YCbCr	4:4:4	Separate	8	24
YCbCr	4:4:4	Separate	10	30
YCbCr	4:4:4	Separate	12	36
YCbCr	4:2:2	Separate	8	16
YCbCr	4:2:2	Separate	10	20

YCbCr	4:2:2	Separate	12	24
YCbCr	4:4:4	Embedded	8	24
YCbCr	4:4:4	Embedded	10	30
YCbCr	4:4:4	Embedded	12	36
YCbCr	4:2:2	Embedded	8	16
YCbCr	4:2:2	Embedded	10	20
YCbCr	4:2:2	Embedded	12	24

## 2. Embedded Sync Extraction Module

The module is used to extract Vsync and Hsync signals from input video data stream such as ITU656 format. With setting the relative registers, this functional module can extract correct video sync signals for later process block using.

## 3. Data Enable (DE) Generator

HDMI Transmitter has DE signal generator by incoming HSYNCs, VSYNCs and Video clock. External DE is optional and selected by appropriate register settings. This feature is particularly useful when interfacing to MPEG decoders that do not provide a specific DE output signal.

## 4. Color Space Conversion

HDMI Transmitter Color space conversion (CSC) is available to interface for several MPEG decoders like with YCbCr-only outputs, and to provide full DVI backwards compatibility.

The function of this module is to perform color space conversion functionality as listed below.

- (1). Convert RGB input Video data to YCbCr Video data.
- (2). Convert YCbCr input Video data to RGB Video data.
- (3). upsample for YCbCr 4:2:2 to YCbCr 4:4:4
- (4). downsample for YCbCr 4:4:4 to YCbCr 4:2:2

### 32.3.2 Audio Data Processing

The HDMI TX audio process contain audio clock regeneration, placement of audio samples within packets, packet timing control, audio sample rates setting, and channel/speaker assignments. This function is implemented by Audio Capture blocks

The Audio Capture support either SPDIF or four channel I2S input. SPDIF input supports audio sampling rates from 32 to 192 KHz. The I2S input supports from 2-channel to 8-channel audio up to 192 KHz.

The scheme of audio processing as shown in the figure below:

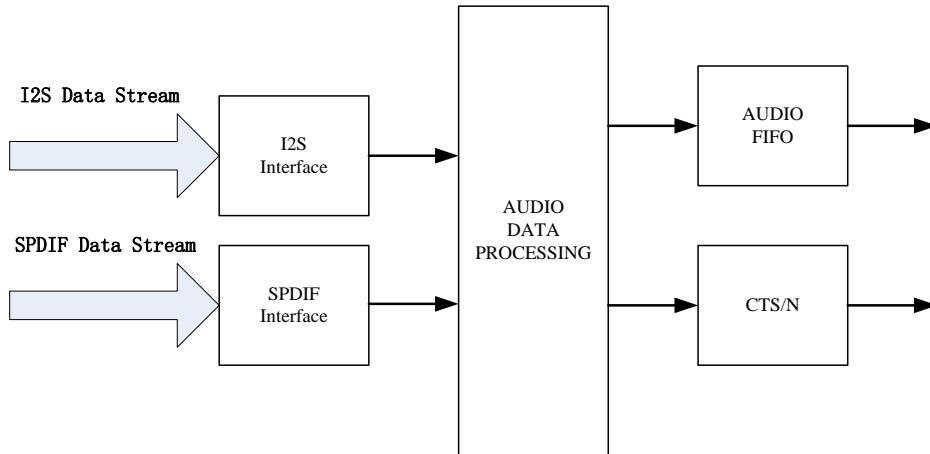


Fig. 32-4 HDMI Audio Data Processing Diagram

### 1.I2S

The function of this module is to implement I2S audio input feature. The incoming audio stream is captured, processed then transmitted into the TMDS link. Four I2S inputs also allow transmission of DVD-Audio and decoded Dolby Digital to A/V Receivers and high-end displays. The interface supports from 2-channel to 8-channel audio up to 192 kHz. The I2S pins must also be coherent with mclk. The appropriate registers must be configured to describe the format of audio being input. This information is passed over the HDMI link in the CEA-861D Audio Info (AI) packets. Table shows the I2S 8 channel audio formats that are supported for each of the video formats.

Table 32-2 HDMI TX I2S 2 Channel Audio Sampling Frequency

Video Format	32kHz	44.1kHz	48kHz	88.2kHz	96kHz	176.4kHz	192kHz
720x480p /720x576p	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1440x480i/ 1440x576i	Yes	Yes	Yes	Yes	Yes	Yes	Yes
720p	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1080i	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1080p	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 32-3 HDMI TX I2S 8 Channel Audio Sampling Frequency

Video Format	32kHz	44.1kHz	48kHz	88.2kHz	96kHz	176.4kHz	192kHz
720x480p /720x576p	Yes	Yes	Yes	No	No	No	No
1440x480i/ 1440x576i	Yes	Yes	Yes	Yes	No	No	No
720p	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1080i	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1080p	Yes	Yes	Yes	Yes	Yes	Yes	Yes

### 2.SPDIF

The function of this module is to implement SPDIF audio input feature. The incoming audio stream is captured, processed then transmitted into the TMDS link. SPDIF stream can carry 2-channel uncompressed PCM data (IEC 60958) or a compressed bit stream for multi-channel (IEC 61937) formats. The audio data capture logic forms the audio data into packets in accordance with the HDMI specification. SPDIF input supports audio sampling rates from 32 to 192 KHz. The following shows the SPDIF audio formats that are supported for each of the video formats

Table 32-4 HDMI SPDIF Sampling Frequency at Each Video Format

Video Format	32kHz	44.1kHz	48kHz	88.2kHz	96kHz	176.4kHz	192kHz
720x480p /720x576p	Yes	Yes	Yes	Yes	Yes	No	No
1440x480i/ 1440x576i	Yes	Yes	Yes	Yes	Yes	No	No
720p	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1080i	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1080p	Yes	Yes	Yes	Yes	Yes	Yes	Yes

### 3.Audio Sample Clock Capture and Regeneration

Audio data being carried across the HDMI link, which is driven by a TMDS clock running at a rate corresponding to the video pixel rate, does not retain the original audio sample clock. The task of recreating this clock at the Sink is called Audio Clock Regeneration.

The HDMI Transmitter determine the fractional relationship between the TMDS clock and an audio reference clock (128 audio sample rate [fs]) and pass the numerator and denominator of that fraction to the HDMI Sink across the HDMI link. The Sink then re-create the audio clock from the TMDS clock by using a clock divider and a clock multiplier.

The exact relationship between the two clocks will be.

$$128 \cdot f_s = f_{TMDS\_clock} \cdot N / CTS.$$

The scheme of the Audio Sample Clock Capture and Regeneration as shown below:

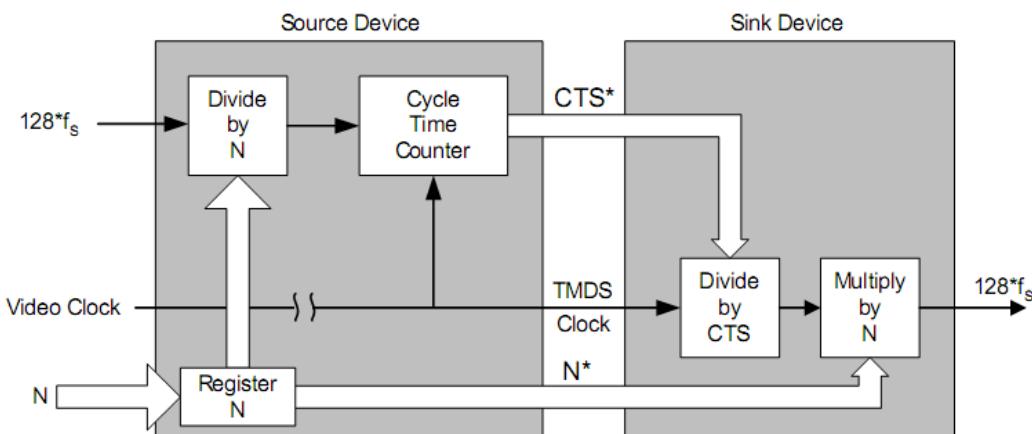


Fig. 32-5 HDMI Audio Clock Regeneration Model

Because there is no audio clock carried through the HDMI link, only the TMDS clock is used. Software sets the CTS/N with a value taken from the below table, which shows the CTS and N value for the supported standard. All other TMDS clocks are not supported; The TMDS clocks divided or multiplied by 1,001 coefficients are not supported.

Table 32-5 HDMI CTS and N table

Fs (kHz)	TMDS Clock (MHz)													
	25.2		27		54		74.25		148.5		297		597	
N	CTS	N	CTS	N	CTS	N	CTS	N	CTS	N	CTS	N	CTS	
32	4096	25200	4096	27000	4096	54000	4096	74250	4096	148500	3072	222750	3072	445500
44.1	6272	28000	6272	30000	6272	60000	6272	82500	6272	165000	4704	247500	9408	990000
48	6144	25200	6144	27000	6144	54000	6144	74250	6144	148500	5120	247500	6144	495000
88.2	12544	28000	12544	30000	12544	60000	12544	82500	12544	165000	9408	247500	18816	990000
96	12288	25200	12288	27000	12288	54000	12288	74250	12288	148500	10240	247500	12288	495000
176.4	25088	28000	25088	30000	25088	60000	25088	82500	25088	165000	18816	247500	37632	990000
192	24576	25200	24576	27000	24576	54000	24576	74250	24576	148500	20480	247500	24576	4950000

### 32.3.3 DDC

The DDC functional block is used for configuration and status exchange between the HDMI Source and HDMI Sink. HDMI Transmitter Controller has I2C Master Interface for DDC transactions. It enables for host controller to read EDID, HDCP authentication by issuing simple register access. The I2C bus speed is limited by DDC specification. DDC bus access frequency can be controlled.

### 32.3.4 EDID

Extended Display Identification Data (EDID) was created by VESA to enable plug and play capabilities of monitors. This data, which is stored in the sink device, describes video formats that the DTV Monitor is capable of receiving and rendering. The information is supplied to the source device, over the interface, upon the request of the source device. The source device then chooses its output format, taking into account the format of the original video stream and the formats supported by the DTV Monitor. The function of this module is to implement EDID feature.

### 32.3.5 HDCP

HDMI Transmitter has a capability for HDCP authentication by hardware. The function of this module is to implement HDCP encryption feature. This feature can be turned on or off depending on register setting.

### 32.3.6 Hot Plug Detect

HDMI Transmitter has a capability for detecting the Sink plug in or plug out, and launch an interrupt and registers state indicating for software controlling.

### 32.3.7 TMDS encoder

The TMDS encoder converts the 2/4/8 bits data into the 10 bit DC-balanced TMDS data.

HDMI TX put the TMDS encoding on the audio /video /aux data received from the HDCP XOR mask. This data is output onto three TMDS differential data lines along with a TMDS differential clock.

### 32.3.8 CEC

The CEC functional block provides high-level control functions between all of the various audiovisual products in a user's environment through one line.

## 32.4 Register Description

The address offset of the HDMI TX is 0xff980000,it contains 16 address section.The offset of the table of Register Summary must multiple with 4 when software configure it. Like the Interrupt registers, its base address is 0x0100.If we want to configure it, its real address is 0xff980000+0x0100\*4.

We can configure the HDMI PHY register through the internal I2C interface. The internal I2C register interface map with the address which from 0x3020. We just to configure the register which can trigger one i2c write or i2c read. For example, we configure the PLL through these register.

### 32.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
Identification Registers	0x0000	B		Identification related registers
Interrupt registers	0x0100	B		Interrupt related registers
Video Sampler registers	0x0200	B		Video Sampler registers
Video Packetizer registers	0x0800	B		Video Packetizer registers
Frame Composer Registers	0x1000	B		Frame Composer Registers
HDMI Source PHY Registers	0x3000	B		HDMI Source PHY Registers
I2C Master PHY Registers	0x3020	B		I2C Master PHY Registers
Audio Sampler Registers	0x3100	B		Audio Sampler Registers
Main Controller Registers	0x4000	B		Main Controller Registers
Color Space Converter Registers	0x4100	B		Color Space Converter Registers
HDCP Encryption Engine Registers	0x5000	B		HDCP Encryption Engine Registers
HDCP BKSV Registers	0x7800	B		HDCP BKSV Registers
HDCP AN Registers	0x7805	B		HDCP AN Registers
Encrypted DPK Embedded Storage Registers	0x780E	B		Encrypted DPK Embedded Storage Registers
CEC Engine Registers	0x7D00	B		CEC Engine Registers
I2C Master Registers	0x7E00	B		I2C Master Registers for E-DDC/SCDC

For the detail register description, please see the doc: [DWC\\_hdmi\\_tx\\_databook.pdf](#) and [dwc\\_hdmi\\_tx\\_ew\\_6gbps-gf28slp18\\_databook\\_rockchip.pdf](#)

## 32.5 Interface Description

### 32.5.1 Video Input Source

In RK3288, the HDMI TX video source comes from VOP\_BIG and VOP\_LIT.

### 32.5.2 Audio Input Source

In RK3288, the HDMI TX audio source comes from I2S\_8CH or SPDIF\_2CH.

## 32.6 Application Notes

This chapter describes how to bring up HDMI transmitter in your system. As shown few examples below, these introduce the basically HDMI transmitter application, likes, the Hot Plug Detect, EDID read back, multiple audio format input and different video resolution displaying.

You can easily configure these functions with proper registers value setting by HDMI TX APB BUS.

### 32.6.1 Initial Operation

The default HDMI transmitter is configured to 24bit RGB 1080P resolution video with 8 channel 48K sample I2S format audio input. It is easily for customer to turn on HDMI transmitter without doing more complex operation. Just do the step, reset the HDMI TX.

### 32.6.2 Hot Plug Detection

Hot Plug Detect is a special feature for HDMI transmitter spying the state on the HDMI port.

You can control this function by using the interrupt signal and proper registers from the HDMI transmitter with few operations. The following is a step by step instruction for detecting the hot plug in and out.

#### **Hot Plug in Steps:**

**Step1:** Write 1'b1 in the phy\_conf0.enhpdrxsense bit field register..

**Step2:** Plug HDMI receiver in.

**Step3:** Check the interrupt from signal pin\_int.

If the pin\_int shows high, that means the HDMI transmitter interrupt have generated.

**Step4:** Check the interrupt.

Read the phy\_stat0.HPD bit field register. If HPD=0,the Hot Plug signal is low(no Sink(Receiver) detected). If HPD=1,the Hot Plug signal is high(Sink(Receiver) detected).

#### **Hot Plug out Steps:**

**Step1:** HDMI transmitter at working state.

**Step2:** Plug HDMI receiver out.

**Step3:** Check the interrupt from signal pin\_int.

If the pin\_int shows high, that means the HDMI transmitter interrupt have generated.

**Step4:** Check the interrupt.

Read the phy\_stat0.HPD bit field register. If HPD=0, the Hot Plug signal is low(no Sink(Receiver) detected). If HPD=1, the Hot Plug signal is high(Sink(Receiver) detected).

### 32.6.3 Reading EDID

Read EDID is a function that can make the HDMI transmitter to read the HDMI receiver's Extended Display Identification Data (EDID) in order to discover the HDMI receiver's configuration and capabilities. HDMI transmitter can choose the appropriate audio and video format for playing and displaying by the HDMI receiver through the use of the EDID. Besides, HDMI transmitter support the reading Enhanced Extended Display Identification Data (E-EDID) if HDMI receiver have this enhanced structure.

The following describes how to read E-EDID through HDMI transmitter. The total E-EDID is 512bytes data, which is divided into 2 segments. Each segment has 256bytes data. The Read E-EDID function is only read 64bytes data from HDMI receiver at each time. So, you must read 8 times that can read total 512bytes data back.

The related registers offset is 0x7E00.

#### Normal read E-EDID 512bytes Steps:

**Step1:** Set I2C slave address.

Write i2cm\_slave.slaveaddr[6:0] bit field register.

**Step2:** Set I2C register address.

Write i2cm\_slave.address[7:0] bit field register.

**Step3:** Activate Sequential Real operation.

Write "1" in the i2cm\_operation.rd8 bit field register.

**Step4:** Wait for interruption

Wait for ii2cmasterdone interrupt in the ih\_i2cm\_stat0 register

**Step5:** Read data result

Read data of registers i2cm\_read\_buff0[7:0] to i2cm\_read\_buff7[7:0]

#### Read E-EDID extended sequential read operation Steps:

**Step1:** Set I2C slave address.

Write i2cm\_slave.slaveaddr[6:0] bit field register.

**Step2:** Set I2C segment address.

Write the i2cm\_segaddr.seg\_addr bit field register.

**Step3:** Set I2C segment pointer.

Write i2cm\_segptr.segptr bit field register.

**Step4:** Activate Read operation.

Write "1" in the i2cm\_operation.rd8\_ext bit field register.

**Step5:** Wait for interruption.

Wait for ii2cmasterdone interrupt in the ih\_i2cm\_stat0 register

**Step6:** Read data result.

Read data of registers i2cm\_read\_buff0[7:0] to i2cm\_read\_buff7[7:0].

### 32.6.4 Audio input configuration

HDMI transmitter audio support either SPDIF or four channel I2S input. SPDIF input supports audio sampling rates from 32 to 192 KHz. The I2S input supports from 2-channel to 8-channel audio up to 192 KHz. The default audio format is I2S input with 8 channels. The audio sample rate is 48K.

The following describes how to configure audio input format. The related register offset is from 0x3100.

#### Configure Audio Input Format with I2S Steps:

**Step1:** Select I2S input.

Write "1" in the aud\_conf0.i2s\_select bit field register.

**Step2:** Enable I2S inputs:

Write "1" in the aud\_conf0.i2s\_in\_en[3:0] bit field register.

**Step3:** Set I2S Mode [Standard | Right-justified | Left-justified | Burst1 | Burst2]:

Write the aud\_conf1.i2s\_mode[2:0] bit field register.

**Step4:** Set I2S data width [16 bits up to 24 bits]:

Write the aud\_conf1.i2s\_width[4:0] bit field.

#### Configure Audio Input Format with SPDIF Steps:

**Step1:** Select SPDIF input.

Write "0" in the aud\_conf0.i2s\_select bit field register.

**Step2:** Set S/PDIF Linear-PCM or Non-Linear PCM audio samples:

Write the aud\_spdif1.setnlpcm bit field register.

**Step3:** Set SPDIF data width [16 bits up to 24 bits]:

Write the aud\_spdif1.spdif\_width[4:0] bit field.

#### Configure Audio Parameters Steps:

**Step1:** Set Audio input frequency clock FS ratio factor [128 Fs | 256 Fs | 512 Fs]:

Write the aud\_inputclkfs.lfsfactor bit field register.

**Step2:** Set Audio fixed N factor for Audio Clock Regeneration. This factor depends on the audio sampling rate and video mode.

Write the aud\_n1.audN, aud\_n2.audN, and aud\_n3.audN bit field registers.

**Step3:** Set Audio CTS factor for Audio Clock Regeneration. This factor can be generated automatically or manually.

For Automatic CTS generation

Write "0" on the bit field "CTS\_manual", Register 0x3205: AUD\_CTS3

For Manual CTS setting

Write "1" in the aud\_cts3.CTS\_manual register bit field.

Write the aud\_cts1.audCTS, aud\_cts2.audCTS, aud\_cts3.audCTS bit field registers.

**Step4:** Enable Audio sampler block:

Write "0" in the mc\_clkdis.audclk\_disable bit field register.

### 32.6.5 Video input configuration

HDMI transmitter support RGB/YCbCr 24/30bit video input with different resolution. The default video format is RGB24bit input at resolution of 1080P@60. The following describes

how to configure video input format into RGB24bit input at resolution of 480P@60, 720P@60 or 1080P@60.

HDMI pin\_vclk cannot get invert.

**Video input requirement:**

24bit RGB 4:4:4 Source.

Resolution is 480P@60, 720P@60 or 1080P@60.

**Configure Video Input Format Steps:**

**Step1:** To select the Video Mapping input mode (RGB444, YCC444, YCC422).

Write the video code in the tx\_invid0.video\_mapping bit field register.

**Step2:** Set video timing information configuration:

Write the fc\_invidconf.vsync\_in\_polarity register.

Write the fc\_invidconf.hsync\_in\_polarity register.

Write the fc\_invidconf.de\_in\_polarity register.

Write the fc\_invidconf.r\_v\_blank\_in\_osc register.

Write the fc\_invidconf.in\_I\_P register.

H active pixels

- Write the fc\_inhactiv1.H\_in\_activ register.

- Write the fc\_inhactiv0.H\_in\_activ register.

V active pixels

- Write the fc\_invactiv1.V\_in\_activ register.

- Write the fc\_invactiv0.V\_in\_activ register.

H blanking pixels

Write the fc\_inhblank0.H\_in\_blank register.

V blanking pixels

Write the fc\_invblank.V\_in\_blank register.

HSync offset

Write the fc\_hsyncindelay0.H\_in\_delay register.

VSync offset

Write the fc\_vsyncindelay0.V\_in\_delay register.

HSync pulse width

Write the fc\_hsyncinwidth0.H\_in\_width register.

VSync pulse width

Write the fc\_vsyncinwidth0.V\_in\_width register.

**Step3:** Select DVI or HDMI mode:

Write "0" for DVI in the fc\_invidconf.DVI\_modez bit field register.

Write "1" for HDMI in the fc\_invidconf.DVI\_modez bit field register.

The detail configuration for AVI information, please refer to the HDMI specification (8.2.1) and CEA-861-D (6.3).

### **32.6.6 HDMI MPLL CONFIGURE**

HDMI transmitter have a PLL for generate the TMDS clock. Configuring the PLL related parameter use the i2c master interface.

### Configure HDMI PLL Step:

**Step1:** Place the PHY in configuration mode by writing 8'h32 to the phy\_conf0 register.

**Step2:** Reset the PHY by writing 0x01 in the mc\_phyrstz register.

**Step3:** Write the desired color depth and the pixel repetition in the vp\_pr\_cd register.

**Step4:** After a PHY-dependent time, it is required to lift the reset by writing 0x00 to the mc\_phyrstz register.

**Step5:** Set the PHY slave address by writing 0x69 in the phy\_i2cm\_slave register.

**Step6:** According to [dwc\\_hdmi\\_tx\\_ew\\_6gbps-gf28slp18\\_databook\\_rockchip.pdf](#), (Appendix B:PLL Configuration) you are required to look up the configuration for your intended video mode and write those values to the PHY I2C interface. The baseline flow to write to the

PHY through the I2C interface is as follows:

- i. Write the register address in the phy\_i2cm\_address register.
- ii. Write data in the phy\_i2cm\_datao\_1 (MSB, [15:8]) and phy\_i2cm\_datao\_0 (LSB, [7:0]) registers.
- iii. Initialize the write operation by writing 8'h10 in the phy\_i2cm\_operation register.
- iv. Wait for a done interruption from the I2C master.

**Step7:** After all of the required PHY I2C registers have been configured, you now need to place the PHY in power-on mode by setting the txpwron bit in the PHY\_CONF0 register, writing 8'h2a to the phy\_conf0 register.

The mc\_phyrstz register controls the PHY reset.

**Step8:** At the end of the PHY configuration, it is recommended to check if the PHY PLL is locked.

Read the phy\_stat0.tx\_phy\_lock bit field register.

If tx\_phy\_lock = 0, the PLL is not locked.

If tx\_phy\_lock = 1, the PLL is locked.

### 32.6.7 CEC OPERATION

The CEC line is used for high-level user control of HDMI-connected devices. The HDMI TX contain CEC TX operations and CEC RX operations.

You can control this function by using the interrupt signal and proper registers from the HDMI transmitter with few operations. The register offset is from 0x7D00.

### Configure The CEC Step:

**Step1:** Write the CEC logical address to cec\_addr\_l, cec\_addr\_h register

**Step2:** Write the size of the frame in bytes which are available in the transmitter data buffer to cec\_tx\_cnt register

**Step3:** Write the desired CEC data(including header and data blocks) to cec\_tx\_data0 to cec\_tx\_data15

**Step4:** Write 1 to cec\_ctrl.send register, to start the cec transmit.

### 32.6.8 HDCP OPERATION

HDCP is designed to protect the transmission of Audiovisual Content between an HDCP Transmitter and an HDCP Receiver. You can control this function by using the interrupt signal and proper registers from the HDMI transmitter with few operations.

The following is a step by step instruction for HDCP operation.

#### HDCP Access KSV Memory Step:

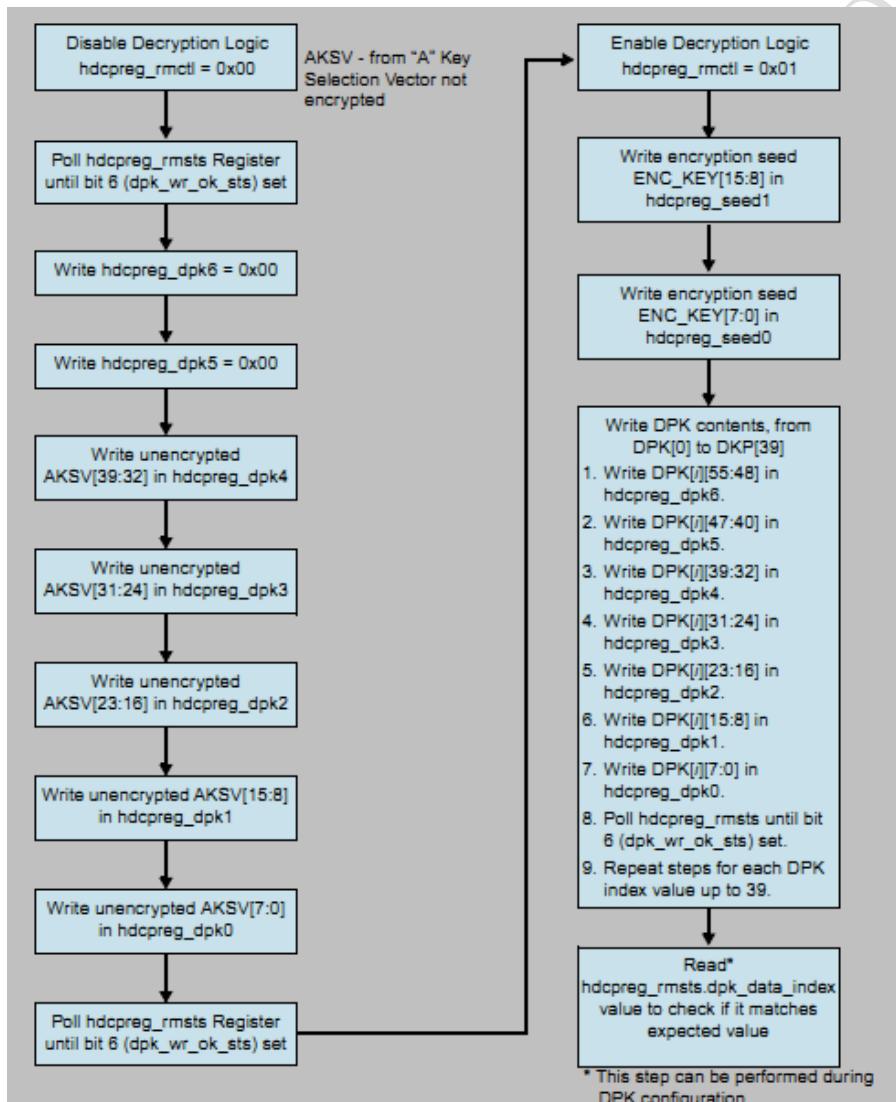
**Step1:** Request access to KSV memory through setting a\_ksvmemctrl.KSVMEMrequest to 1'b1 and pool

a\_ksvmemctrl.KSVMEMaccess until this value is 1'b1 (access granted).

**Step2:** Read VH', M0, Bstatus, and the KSV FIFO.

The data is stored in the revocation memory, as provided in the “Address Mapping for Maximum Memory Allocation” table in the [hdmi\\_databook](#).

#### HDCP Key Write Step:



## Chapter 33 LVDS

### 33.1 Overview

LVDS transmitter converts a CMOS signal into a low-voltage differential signal. Using a differential signal reduces the system's susceptibility to noise and EMI emissions. In addition, using a differential signal can deliver high speeds. This results in a very cost-effective solution to some of the greatest bandwidth bottlenecks in many transmission applications.

LVDS supports following features:

- Comply with the TIA/EIA-644-A LVDS standard
- Combine LVTTL IO, support LVDS/LVTTL data output
- Support reference clock frequency range from 10Mhz to 148.5Mhz
- Support LVDS RGB 30/24/18bits color data transfer
- Support VESA/JEIDA LVDS data format transfer
- Support LVDS single channel and double channel data transfer, every channel include 4 data lanes and 1 clock lane
- Support MSB mode and LSB mode data transfer
- Support APB slave bus interface
- Support low power mode

### 33.2 Block Diagram

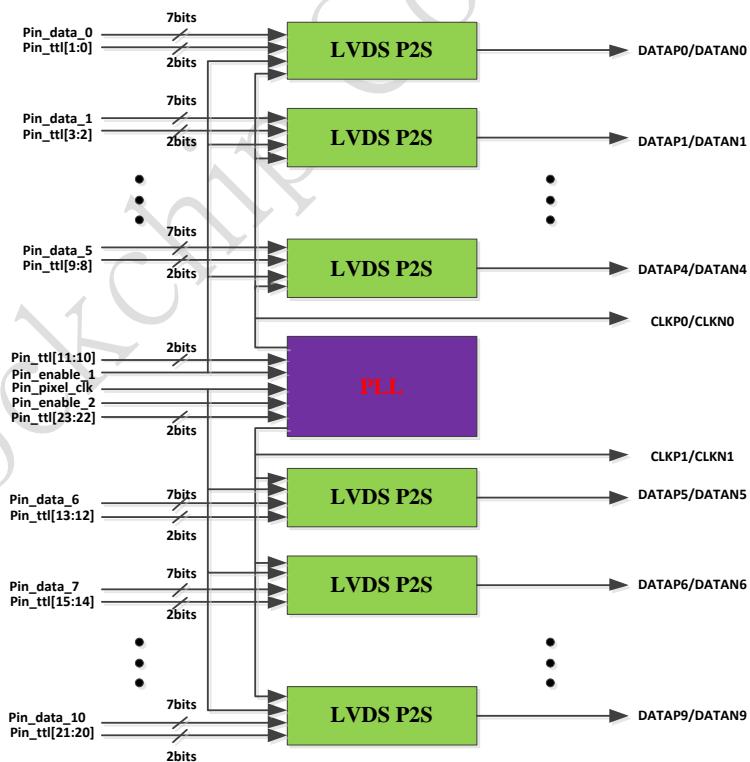


Fig. 33-1 LVDS Block Diagram

Fig.1-1 shows the brief block diagram of the innosilicon LVDS/TTL PHY, which includes ten LVDS P2S modules and one PLL module.

PLL is responsible for multiplying the pin\_pixel\_clk by 7, which generates a 7X clock used to deserialize the parallel data.

LVDS P2S module implements the parallel to serial function and transmits the TTL data directly.

### 33.3 Function Description

#### 33.3.1 Transmitter with Two 35:5 Data Channels

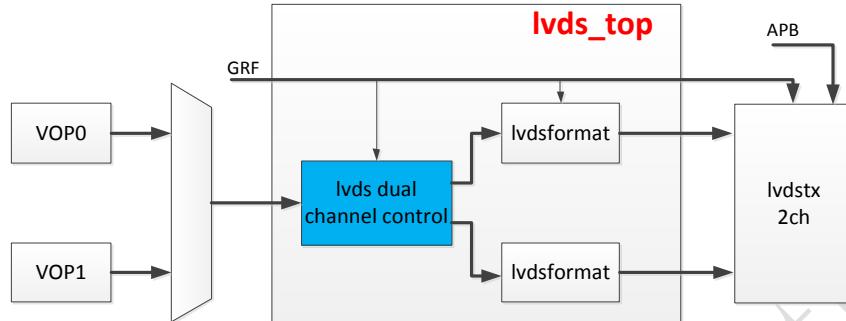


Fig. 33-2 LVDS in SoC

There are two transfer channels in LVDS, every channel include 4 data lanes and 1 clock lane. LVDS can work at single channel mode or double channel mode.

The LVDS output data timing is showed as the following figure,

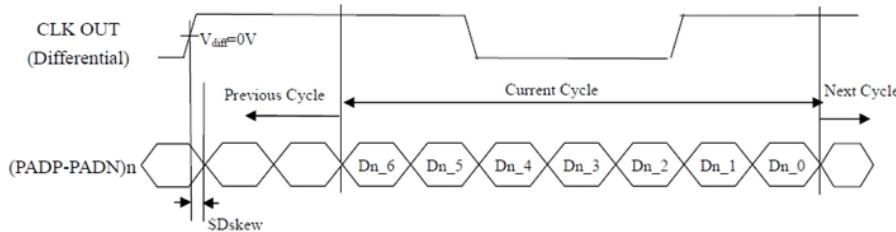


Fig. 33-3 LVDS output data timing

#### 33.3.2 LVDS Format

Lvdsformat converts VOP RGB interface to LVDS format data. The lvdsformat support RGB 10/8/6bits color data input. There are two lvdsformat modules in the lvds\_top.

When LVDS works at double channel mode, both the lvdsformat modules are necessary. One is for odd cycle RGB data format convert, and the other is for even cycle RGB data format convert. The frequency of LVDS output clock is half of the dclk.

When LVDS works at single channel mode, only one lvdsformat module is necessary. User can configure GRF register to select which lvdsformat converts the VOP RGB data. The frequency of LVDS output clock is equal to the dclk.

Table 33-1 is the MSB mapping relationship between the input data and output data of lvdsformat module (single channel mode). The LSB mapping relationship is opposite to MSB.

Table 33-1 MSB mapping relationship (single channel mode)

Lvds-format id	Serial Channel	Data Bits	RGB10 Bits		RGB8 Bits			RGB6 bits
			format-1	format-2	format-1	format-2	format-3	
Lvds-format *	DATA0	DATA0[0]	R0	R4	R0	R2	R2	R0
		DATA0[1]	R1	R5	R1	R3	R3	R1

		DATA0[2]	R2	R6	R2	R4	R4	R2
		DATA0[3]	R3	R7	R3	R5	R5	R3
		DATA0[4]	R4	R8	R4	R6	R6	R4
		DATA0[5]	R5	R9	R5	R7	R7	R5
		DATA0[6]	G0	G4	G0	G2	G2	G0
	DATA1	DATA1[0]	G1	G5	G1	G3	G3	G1
		DATA1[1]	G2	G6	G2	G4	G4	G2
		DATA1[2]	G3	G7	G3	G5	G5	G3
		DATA1[3]	G4	G8	G4	G6	G6	G4
		DATA1[4]	G5	G9	G5	G7	G7	G5
		DATA1[5]	B0	B4	B0	B2	B2	B0
		DATA1[6]	B1	B5	B1	B3	B3	B1
	DATA2	DATA2[0]	B2	B6	B2	B4	B4	B2
		DATA2[1]	B3	B7	B3	B5	B5	B3
		DATA2[2]	B4	B8	B4	B6	B6	B4
		DATA2[3]	B5	B9	B5	B7	B7	B5
		DATA2[4]	HSYNC	HSYNC	HSYNC	HSYNC	HSYNC	HSYNC
		DATA2[5]	VSYNC	VSYNC	VSYNC	VSYNC	VSYNC	VSYNC
		DATA2[6]	DEN	DEN	DEN	DEN	DEN	DEN
	DATA3	DATA3[0]	R6	R2	R6	R0	GND	GND
		DATA3[1]	R7	R3	R7	R1	GND	GND
		DATA3[2]	G6	G2	G6	G0	GND	GND
		DATA3[3]	G7	G3	G7	G1	GND	GND
		DATA3[4]	B6	B2	B6	B0	GND	GND
		DATA3[5]	B7	B3	B7	B1	GND	GND
		DATA3[6]	GND	GND	GND	GND	GND	GND
	DATA4	DATA4[0]	R8	R0	GND	GND	GND	GND
		DATA4[1]	R9	R1	GND	GND	GND	GND
		DATA4[2]	G8	G0	GND	GND	GND	GND
		DATA4[3]	G9	G1	GND	GND	GND	GND
		DATA4[4]	B8	B0	GND	GND	GND	GND
		DATA4[5]	B9	B1	GND	GND	GND	GND
		DATA4[6]	GND	GND	GND	GND	GND	GND
	CLKOUT	CLKOUT	DCLK	DCLK	DCLK	DCLK	DCLK	DCLK

Table 33-2 is the MSB mapping relationship between the input data and output data of lvdsformat module (double channel mode). The LSB mapping relationship is opposite to MSB.

Table 33-2 MSB mapping relationship (double channel mode)

lvds-format id	Serial Channel	Data Bits	RGB10 Bits		RGB8 Bits			RGB6 bits
			format-1	format-2	format-1	format-2	format-3	
lvds-format 0	DATA0	DATA0[0]	OR0	OR4	OR0	OR2	OR2	OR0
		DATA0[1]	OR1	OR5	OR1	OR3	OR3	OR1
		DATA0[2]	OR2	OR6	OR2	OR4	OR4	OR2

		DATA0[3]	OR3	OR7	OR3	OR5	OR5	OR3
		DATA0[4]	OR4	OR8	OR4	OR6	OR6	OR4
		DATA0[5]	OR5	OR9	OR5	OR7	OR7	OR5
		DATA0[6]	OG0	OG4	OG0	OG2	OG2	OG0
DATA1		DATA1[0]	OG1	OG5	OG1	OG3	OG3	OG1
		DATA1[1]	OG2	OG6	OG2	OG4	OG4	OG2
		DATA1[2]	OG3	OG7	OG3	OG5	OG5	OG3
		DATA1[3]	OG4	OG8	OG4	OG6	OG6	OG4
		DATA1[4]	OG5	OG9	OG5	OG7	OG7	OG5
		DATA1[5]	OB0	OB4	OB0	OB2	OB2	OB0
		DATA1[6]	OB1	OB5	OB1	OB3	OB3	OB1
DATA2		DATA2[0]	OB2	OB6	OB2	OB4	OB4	OB2
		DATA2[1]	OB3	OB7	OB3	OB5	OB5	OB3
		DATA2[2]	OB4	OB8	OB4	OB6	OB6	OB4
		DATA2[3]	OB5	OB9	OB5	OB7	OB7	OB5
		DATA2[4]	H SYNC					
		DATA2[5]	V SYNC					
		DATA2[6]	DEN	DEN	DEN	DEN	DEN	DEN
DATA3		DATA3[0]	OR6	OR2	OR6	OR0	GND	GND
		DATA3[1]	OR7	OR3	OR7	OR1	GND	GND
		DATA3[2]	OG6	OG2	OG6	OG0	GND	GND
		DATA3[3]	OG7	OG3	OG7	OG1	GND	GND
		DATA3[4]	OB6	OB2	OB6	OB0	GND	GND
		DATA3[5]	OB7	OB3	OB7	OB1	GND	GND
		DATA3[6]	GND	GND	GND	GND	GND	GND
DATA4		DATA4[0]	OR8	OR0	GND	GND	GND	GND
		DATA4[1]	OR9	OR1	GND	GND	GND	GND
		DATA4[2]	OG8	OG0	GND	GND	GND	GND
		DATA4[3]	OG9	OG1	GND	GND	GND	GND
		DATA4[4]	OB8	OB0	GND	GND	GND	GND
		DATA4[5]	OB9	OB1	GND	GND	GND	GND
		DATA4[6]	GND	GND	GND	GND	GND	GND
	CLKOUT	CLKOUT	DCLK/2	DCLK/2	DCLK/2	DCLK/2	DCLK/2	DCLK/2
lvds-format 1	DATA0	DATA0[0]	ER0	ER4	ER0	ER2	ER2	ER0
		DATA0[1]	ER1	ER5	ER1	ER3	ER3	ER1
		DATA0[2]	ER2	ER6	ER2	ER4	ER4	ER2
		DATA0[3]	ER3	ER7	ER3	ER5	ER5	ER3
		DATA0[4]	ER4	ER8	ER4	ER6	ER6	ER4
		DATA0[5]	ER5	ER9	ER5	ER7	ER7	ER5
		DATA0[6]	EG0	EG4	EG0	EG2	EG2	EG0
DATA1		DATA1[0]	EG1	EG5	EG1	EG3	EG3	EG1
		DATA1[1]	EG2	EG6	EG2	EG4	EG4	EG2
		DATA1[2]	EG3	EG7	EG3	EG5	EG5	EG3
		DATA1[3]	EG4	EG8	EG4	EG6	EG6	EG4
		DATA1[4]	EG5	EG9	EG5	EG7	EG7	EG5

	DATA1[5]	EB0	EB4	EB0	EB2	EB2	EB0
	DATA1[6]	EB1	EB5	EB1	EB3	EB3	EB1
DATA2	DATA2[0]	EB2	EB6	EB2	EB4	EB4	EB2
	DATA2[1]	EB3	EB7	EB3	EB5	EB5	EB3
	DATA2[2]	EB4	EB8	EB4	EB6	EB6	EB4
	DATA2[3]	EB5	EB9	EB5	EB7	EB7	EB5
	DATA2[4]	H SYNC					
	DATA2[5]	V SYNC					
	DATA2[6]	DEN	DEN	DEN	DEN	DEN	DEN
	DATA3[0]	ER6	ER2	ER6	ER0	GND	GND
DATA3	DATA3[1]	ER7	ER3	ER7	ER1	GND	GND
	DATA3[2]	EG6	EG2	EG6	EG0	GND	GND
	DATA3[3]	EG7	EG3	EG7	EG1	GND	GND
	DATA3[4]	EB6	EB2	EB6	EB0	GND	GND
	DATA3[5]	EB7	EB3	EB7	EB1	GND	GND
	DATA3[6]	GND	GND	GND	GND	GND	GND
	DATA4[0]	ER8	ER0	GND	GND	GND	GND
DATA4	DATA4[1]	ER9	ER1	GND	GND	GND	GND
	DATA4[2]	EG8	EG0	GND	GND	GND	GND
	DATA4[3]	EG9	EG1	GND	GND	GND	GND
	DATA4[4]	EB8	EB0	GND	GND	GND	GND
	DATA4[5]	EB9	EB1	GND	GND	GND	GND
	DATA4[6]	GND	GND	GND	GND	GND	GND

### 33.3.3 GRF Relative Register Description

GRF\_SOC\_CON6[3] (grf\_con\_lvds\_lcdc\_sel):

- 1'b0: lvds video source from vop0;
- 1'b1: lvds video source from vop1;

GRF\_SOC\_CON7[2:0] (grf\_lvds\_con\_select):

- 3'b000: select RGB8 bits format-1;
- 3'b001: select RGB8 bits format-2;
- 3'b010: select RGB8 bits format-3;
- 3'b011: select RGB6 bits format;
- 3'b100: select RGB10 bits format-1;
- 3'b101: select RGB10 bits format-2;

GRF\_SOC\_CON7[3] (grf\_lvds\_con\_msbsel):

- 1'b0: LSB for lvdsformat;
- 1'b1: MSB for lvds format;

GRF\_SOC\_CON7[4] (grf\_lvds\_con\_chasel):

- 1'b0: single channel mode;
- 1'b1: double channel mode;

GRF\_SOC\_CON7[5] (grf\_lvds\_con\_startsel):

- 1'b0: when lvds works at single channel mode, select lvdsformat 0 for RGB data convert;
- 1'b1: when lvds works at single channel mode, select lvdsformat 1 for RGB data convert;

GRF\_SOC\_CON7[6] (grf\_lvds\_con\_ttl\_en):

- 1'b0: disable lvds ttl mode;
- 1'b1: enable lvds ttl mode;

GRF\_SOC\_CON7[7] (grf\_lvds\_con\_startphase):

- 1'b0: dclk\_div2 start phase reset to 0 at beginning of hs;
- 1'b1: dclk\_div2 start phase reset to 1 at beginning of hs;

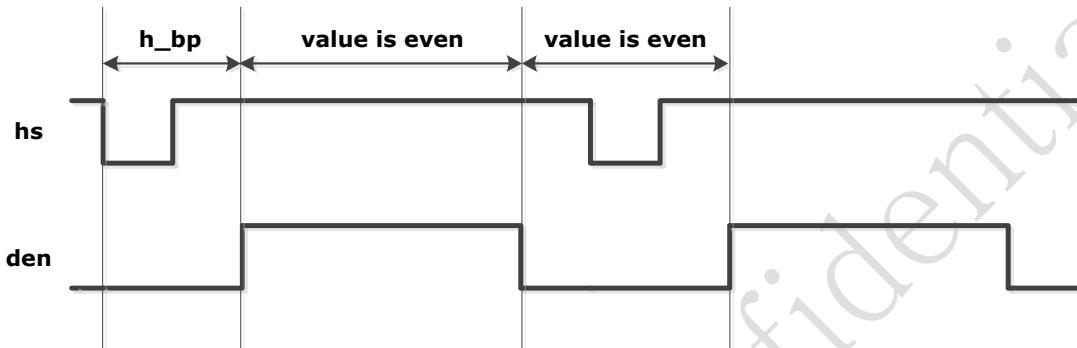


Fig. 33-4 LVDS h\_bp timing diagram

when h\_bp is odd, grf\_lvds\_con\_startphase need be configured to 1'b1;

when h\_bp is even, grf\_lvds\_con\_startphase need be configured to 1'b0;

GRF\_SOC\_CON7[8] (grf\_lvds\_con\_clkinv):

- 1'b0: not invert the clock to LVDS from lvds\_top;
- 1'b1: invert the clock to LVDS from lvds\_top;

GRF\_SOC\_CON7[9] (grf\_lvds\_con\_hs\_polarity):

- 1'b0: hsync polarity low active;
- 1'b1: hsync polarity high active;

GRF\_SOC\_CON7[10] (grf\_lvds\_con\_den\_polarity):

- 1'b0: den polarity high active;
- 1'b1: den polarity low active;

GRF\_SOC\_CON7[11] (grf\_lvds\_con\_enable\_1):

- 1'b0: LVDS channel 1 disable;
- 1'b1: LVDS channel 1 enable;

GRF\_SOC\_CON7[12] (grf\_lvds\_con\_enable\_2):

- 1'b0: LVDS channel 2 disable;
- 1'b1: LVDS channel 2 enable;

GRF\_SOC\_CON7[15] (grf\_lvds\_pwrdown):

- 1'b0: LVDS not power down;

- 1'b1: LVDS power down;

## 33.4 Register Description

This section describes the control/status registers of the design.

### 33.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
LVDS_channel0_reg0	0x0000	W	0x000000bf	LVDS register
LVDS_channel0_reg01	0x0004	W	0x0000003f	LVDS register
LVDS_channel0_reg02	0x0008	W	0x0000007e	LVDS register
LVDS_channel0_reg03	0x000c	W	0x00000046	LVDS register
LVDS_channel0_reg04	0x0010	W	0x00000000	LVDS register
LVDS_channel0_reg05	0x0014	W	0x00000000	LVDS register
LVDS_config_reg0c	0x0030	W	0x00000000	LVDS register
LVDS_channel0_reg0d	0x0034	W	0x0000000a	LVDS register
LVDS_channel0_reg20	0x0080	W	0x00000045	LVDS register
LVDS_config_reg21	0x0084	W	0x00000000	LVDS register
LVDS_channel1_reg40	0x0100	W	0x000000bf	LVDS register
LVDS_channel1_reg41	0x0104	W	0x0000003f	LVDS register
LVDS_channel1_reg42	0x0108	W	0x0000007e	LVDS register
LVDS_channel1_reg43	0x010c	W	0x00000046	LVDS register
LVDS_channel1_reg44	0x0110	W	0x00000000	LVDS register
LVDS_channel1_reg45	0x0114	W	0x00000000	LVDS register
LVDS_channel1_reg4d	0x0134	W	0x0000000a	LVDS register
LVDS_channel1_reg60	0x0180	W	0x00000045	LVDS register

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

### 33.4.2 Detail Register Description

#### LVDS\_channel0\_reg00

Address: Operational Base + offset (0x0000)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7	RW	0x1	lvds_mode_en 1'b1: enable lvds mode; 1'b0: disable lvds mode;
6	RW	0x0	ttl_mode_en 1'b1: enable ttl mode; 1'b0: disable ttl mode;
5	RW	0x1	lane_en_ck 1'b1: enable lane_ck; 1'b0: disable lane_ck;
4	RW	0x1	lane_en_4 1'b1: enable lane_4; 1'b0: disable lane_4;
3	RW	0x1	lane_en_3 1'b1: enable lane_3; 1'b0: disable lane_3;
2	RW	0x1	lane_en_2 1'b1: enable lane_2; 1'b0: disable lane_2;
1	RW	0x1	lane_en_1 1'b1: enable lane_1; 1'b0: disable lane_1;
0	RW	0x1	lane_en_0 1'b1: enable lane_0; 1'b0: disable lane_0;

#### LVDS\_channel0\_reg01

Address: Operational Base + offset (0x0004)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x1	biasen_ck 1'b1: enable lane_ck bias; 1'b0: disable lane_ck bias;
4	RW	0x1	biasen_4 1'b1: enable lane_4 bias; 1'b0: disable lane_4 bias;
3	RW	0x1	biasen_3 1'b1: enable lane_3 bias; 1'b0: disable lane_3 bias;

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x1	biasen_2 1'b1: enable lane_2 bias; 1'b0: disable lane_2 bias;
1	RW	0x1	biasen_1 1'b1: enable lane_1 bias; 1'b0: disable lane_1 bias;
0	RW	0x1	biasen_0 1'b1: enable lane_0 bias; 1'b0: disable lane_0 bias;

**LVDS\_channel0\_reg02**

Address: Operational Base + offset (0x0008)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6	RW	0x1	lane_lvds_en_ck 1'b1: enable lane_ck lvds mode; 1'b0: disable lane_ck lvds mode;
5	RW	0x1	lane_lvds_en_4 1'b1: enable lane_4 lvds mode; 1'b0: disable lane_4 lvds mode;
4	RW	0x1	lane_lvds_en_3 1'b1: enable lane_3 lvds mode; 1'b0: disable lane_3 lvds mode;
3	RW	0x1	lane_lvds_en_2 1'b1: enable lane_2 lvds mode; 1'b0: disable lane_2 lvds mode;
2	RW	0x1	lane_lvds_en_1 1'b1: enable lane_1 lvds mode; 1'b0: disable lane_1 lvds mode;
1	RW	0x1	lane_lvds_en_0 1'b1: enable lane_0 lvds mode; 1'b0: disable lane_0 lvds mode;
0	RW	0x0	pll_fbdv_8 pll_fbdv[8];

**LVDS\_channel0\_reg03**

Address: Operational Base + offset (0x000c)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x46	pll_fbdv_7_to_0 pll_fbdv[7:0];

**LVDS\_channel0\_reg04**

Address: Operational Base + offset (0x0010)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x0	lane_ttl_en_ck 1'b1: enable lane_ck ttl mode; 1'b0: disable lane_ck ttl mode;
4	RW	0x0	lane_ttl_en_4 1'b1: enable lane_4 ttl mode; 1'b0: disable lane_4 ttl mode;
3	RW	0x0	lane_ttl_en_3 1'b1: enable lane_3 ttl mode; 1'b0: disable lane_3 ttl mode;
2	RW	0x0	lane_ttl_en_2 1'b1: enable lane_2 ttl mode; 1'b0: disable lane_2 ttl mode;
1	RW	0x0	lane_ttl_en_1 1'b1: enable lane_1 ttl mode; 1'b0: disable lane_1 ttl mode;
0	RW	0x0	lane_ttl_en_0 1'b1: enable lane_0 ttl mode; 1'b0: disable lane_0 ttl mode;

**LVDS\_channel0\_reg05**

Address: Operational Base + offset (0x0014)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x0	lane_ttl_ctrl_ck 1'b1: enable lane_ck ttl data transmission; 1'b0: disable lane_ck ttl data transmission;
4	RW	0x0	lane_ttl_ctrl_4 1'b1: enable lane_4 ttl data transmission; 1'b0: disable lane_4 ttl data transmission;
3	RW	0x0	lane_ttl_ctrl_3 1'b1: enable lane_3 ttl data transmission; 1'b0: disable lane_3 ttl data transmission;
2	RW	0x0	lane_ttl_ctrl_2 1'b1: enable lane_2 ttl data transmission; 1'b0: disable lane_2 ttl data transmission;
1	RW	0x0	lane_ttl_ctrl_1 1'b1: enable lane_1 ttl data transmission; 1'b0: disable lane_1 ttl data transmission;
0	RW	0x0	lane_ttl_ctrl_0 1'b1: enable lane_0 ttl data transmission; 1'b0: disable lane_0 ttl data transmission;

**LVDS\_config\_reg0c**

Address: Operational Base + offset (0x0030)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x00	enable_pll 8'h00: enable pll;

**LVDS\_channel0\_reg0d**

Address: Operational Base + offset (0x0034)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RW	0x0a	pll_prediv_4_to_0 pll_prediv[4:0];

**LVDS\_channel0\_reg20**

Address: Operational Base + offset (0x0080)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x45	msb_lsb_sel 8'h45: MSB; 8'h44: LSB;

**LVDS\_config\_reg21**

Address: Operational Base + offset (0x0084)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x00	enable_tx 8'h00: disable tx; 8'h92: enable tx;

**LVDS\_channel1\_reg40**

Address: Operational Base + offset (0x0100)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7	RW	0x1	lvds_mode_en 1'b1: enable lvds mode; 1'b0: disable lvds mode;

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RW	0x0	ttl_mode_en 1'b1: enable ttl mode; 1'b0: disable ttl mode;
5	RW	0x1	lane_en_ck 1'b1: enable lane_ck; 1'b0: disable lane_ck;
4	RW	0x1	lane_en_4 1'b1: enable lane_4; 1'b0: disable lane_4;
3	RW	0x1	lane_en_3 1'b1: enable lane_3; 1'b0: disable lane_3;
2	RW	0x1	lane_en_2 1'b1: enable lane_2; 1'b0: disable lane_2;
1	RW	0x1	lane_en_1 1'b1: enable lane_1; 1'b0: disable lane_1;
0	RW	0x1	lane_en_0 1'b1: enable lane_0; 1'b0: disable lane_0;

**LVDS\_channel1\_reg41**

Address: Operational Base + offset (0x0104)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x1	biasen_ck 1'b1: enable lane_ck bias; 1'b0: disable lane_ck bias;
4	RW	0x1	biasen_4 1'b1: enable lane_4 bias; 1'b0: disable lane_4 bias;
3	RW	0x1	biasen_3 1'b1: enable lane_3 bias; 1'b0: disable lane_3 bias;
2	RW	0x1	biasen_2 1'b1: enable lane_2 bias; 1'b0: disable lane_2 bias;
1	RW	0x1	biasen_1 1'b1: enable lane_1 bias; 1'b0: disable lane_1 bias;
0	RW	0x1	biasen_0 1'b1: enable lane_0 bias; 1'b0: disable lane_0 bias;

**LVDS\_channel1\_reg42**

Address: Operational Base + offset (0x0108)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6	RW	0x1	lane_lvds_en_ck 1'b1: enable lane_ck lvds mode; 1'b0: disable lane_ck lvds mode;
5	RW	0x1	lane_lvds_en_4 1'b1: enable lane_4 lvds mode; 1'b0: disable lane_4 lvds mode;
4	RW	0x1	lane_lvds_en_3 1'b1: enable lane_3 lvds mode; 1'b0: disable lane_3 lvds mode;
3	RW	0x1	lane_lvds_en_2 1'b1: enable lane_2 lvds mode; 1'b0: disable lane_2 lvds mode;
2	RW	0x1	lane_lvds_en_1 1'b1: enable lane_1 lvds mode; 1'b0: disable lane_1 lvds mode;
1	RW	0x1	lane_lvds_en_0 1'b1: enable lane_0 lvds mode; 1'b0: disable lane_0 lvds mode;
0	RW	0x0	pll_fbdv_8 pll_fbdv[8];

**LVDS\_channel1\_reg43**

Address: Operational Base + offset (0x010c)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x46	pll_fbdv_7_to_0 pll_fbdv[7:0];

**LVDS\_channel1\_reg44**

Address: Operational Base + offset (0x0110)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x0	lane_ttl_en_ck 1'b1: enable lane_ck ttl mode; 1'b0: disable lane_ck ttl mode;
4	RW	0x0	lane_ttl_en_4 1'b1: enable lane_4 ttl mode; 1'b0: disable lane_4 ttl mode;

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	lane_ttl_en_3 1'b1: enable lane_3 ttl mode; 1'b0: disable lane_3 ttl mode;
2	RW	0x0	lane_ttl_en_2 1'b1: enable lane_2 ttl mode; 1'b0: disable lane_2 ttl mode;
1	RW	0x0	lane_ttl_en_1 1'b1: enable lane_1 ttl mode; 1'b0: disable lane_1 ttl mode;
0	RW	0x0	lane_ttl_en_0 1'b1: enable lane_0 ttl mode; 1'b0: disable lane_0 ttl mode;

**LVDS\_channel1\_reg45**

Address: Operational Base + offset (0x0114)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x0	lane_ttl_ctr_ck 1'b1: enable lane_ck ttl data transmission; 1'b0: disable lane_ck ttl data transmission;
4	RW	0x0	lane_ttl_ctr_4 1'b1: enable lane_4 ttl data transmission; 1'b0: disable lane_4 ttl data transmission;
3	RW	0x0	lane_ttl_ctr_3 1'b1: enable lane_3 ttl data transmission; 1'b0: disable lane_3 ttl data transmission;
2	RW	0x0	lane_ttl_ctr_2 1'b1: enable lane_2 ttl data transmission; 1'b0: disable lane_2 ttl data transmission;
1	RW	0x0	lane_ttl_ctr_1 1'b1: enable lane_1 ttl data transmission; 1'b0: disable lane_1 ttl data transmission;
0	RW	0x0	lane_ttl_ctr_0 1'b1: enable lane_0 ttl data transmission; 1'b0: disable lane_0 ttl data transmission;

**LVDS\_channel1\_reg4d**

Address: Operational Base + offset (0x0134)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RW	0x0a	pll_pdiv_4_to_0 pll_pdiv[4:0];

**LVDS\_channel1\_reg60**

Address: Operational Base + offset (0x0180)

LVDS register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x45	msb_lsb_sel 8'h45: MSB; 8'h44: LSB;

**33.5 Interface Description**

In RK3288, the LVDS video source comes from vop0 or vop1.

- GRF\_SOC\_CON6[3] == 1'b0, video source from vop0.
- GRF\_SOC\_CON6[3] == 1'b1, video source from vop1.

**33.6 Application Notes**

Following is the operation flow which describes how the software configures the registers to start lvds data transmission.

- Select the video source by GRF register;
- Select single channel mode or double channel mode by GRF register;
- Select data transfer format by GRF register;
- Enable lvds pll by LVDS registers;
- Enable lvds transfer by LVDS registers;

## Chapter 34 eDP TX Controller

### 34.1 Overview

This eDP TX IP is compliant with DisplayPort standard 1.2a and eDP 1.3. DisplayPort is an industry standard to accommodate the growing broad adoption of digital display technology within the PC and consumer electronics (CE) industries. It consolidates the internal and external connection methods to reduce device complexity and cost, supports necessary features for key cross industry applications, and provides performance scalability to enable the next generation of displays featuring higher color depths, refresh rates, and display resolutions.

This DisplayPort 1.2 specification defines a scalable digital display interface with optional content protection capability for broad application within PC and CE devices. The interface is designed to support both internal chip-to-chip and external box-to-box digital display connections. Potential internal chip-to-chip applications include usage within a notebook PC for driving a panel from a graphics controller, and usage within a monitor or TV for driving the display component from a display controller. Examples of box-to-box applications for DisplayPort include display connections between PCs and monitors, projectors, and TV displays. DisplayPort is also suitable for display connections between consumer electronics devices such as high definition optical disc players, set top boxes, and TV displays.

It supports following features:

- Compliant with DisplayPort™ Specification, Version 1.2.
- Compliant with eDP™ Specification, Version 1.3.
- HDCP v1.3 amendment for DisplayPort™ Revision 1.0.
- Main link containing 4 physical lanes of 2.7/1.62 Gbps/lane
- TX PHY lanes, control pins and hot-plug pins are shared by the DisplayPort Source
- Bi-directional auxiliary link with up to 1Mbps speed.
- RGB, YCbCr 4:4:4, YCbCr 4:2:2 and 8/10/12 bit per component video format.
- Video and audio slave mode
- Support PSR
- I2S audio interface
- 2,4,6,8-ch PCM - IEC60958 compliant
- S/PDIF audio interface
- Encoded bit stream (Dolby Digital, or DTS) – IEC61937 compliant
- APB slave bus interface
- Hot plug and unplug detection and link status monitor.
- Support VESA DMT and CTV timing standards.
- Fully support EIA/CEA-861Dvideo timing and Info Frame structure.
- Supports reading of the display EDID whenever the display is connected to power, even an AC-trickle power.
- Up to 0.5% down-spreading support at high-speed link.
- Supports DDC/CI and MCCS command transmission when the monitor includes a display controller.
- Flexible output channel mapping and polarity setting.
- PRBS or programmable transmitter pattern for main link quality test.
- Integrated HDCP encryption engine for transmitting protected audio and video content
- SPSRAM interface to read external encrypted HDCP key
- 24 Mhz crystal clock input.

- Built-in video and audio BIST patterns.
- 28nm LP CMOS process with Core voltage 0.9V (min)/ 1.0V (typ)/ 1.08V (max) @ global corner.

## 34.2 Block Diagram

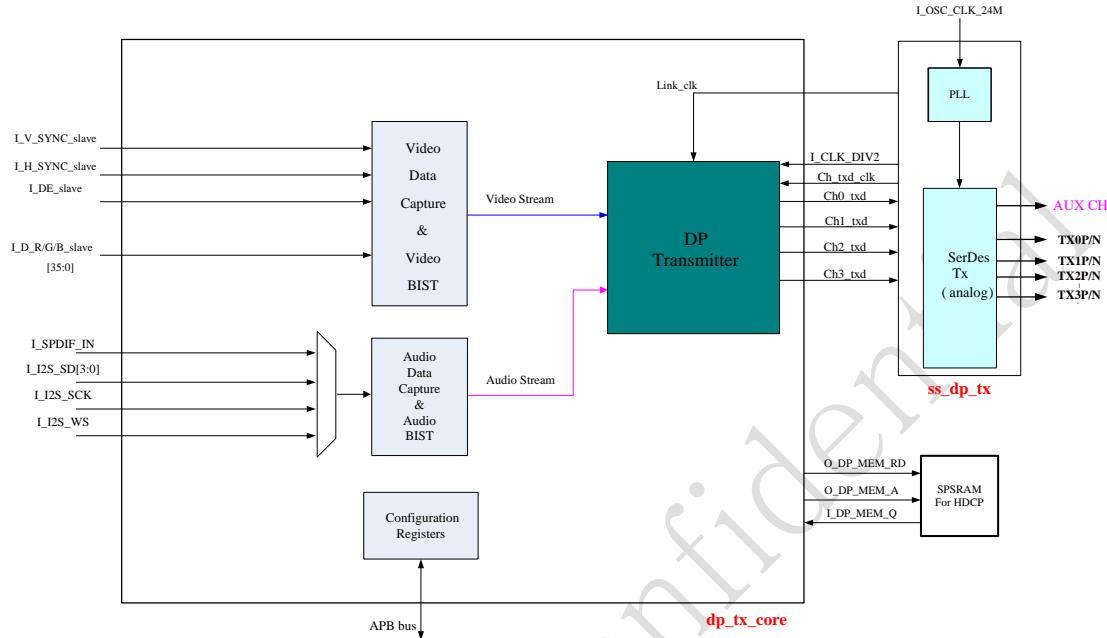


Fig. 34-1 eDP TX controller Block Diagram

Fig.1-1 shows the block diagram of eDP TX controller in top level. The video data and clock are sent directly from the VOP0 or VOP1.

The audio input has 2 interfaces, SPDIF and I2S.

The video data capture & video BIST block is separated as video\_capture and display\_bist module. The audio data capture & audio BIST block is separated as audio\_capture and audio\_bist. The block before SerDes is DP\_TX main module. Following Table shows the brief function description of each sub-module.

Table 34-1 Brief function description of each module in top level

Module Name in Top Level	Brief Module Function Description
video_capture	Capture block of video data.
display_bist	Generation of arbitrary video format with three types of video data. The output of display_bist module will input to video_capture module directly if display BIST mode is active.
audio_capture	Capture block of audio data.
audio_bist	Generation of the audio BIST pattern.
dp_tx	DisplayPort transmitter block.
apb_slave_top	APB slave Bus interface

## 34.3 Function Description

### 34.3.1 eDP in SoC

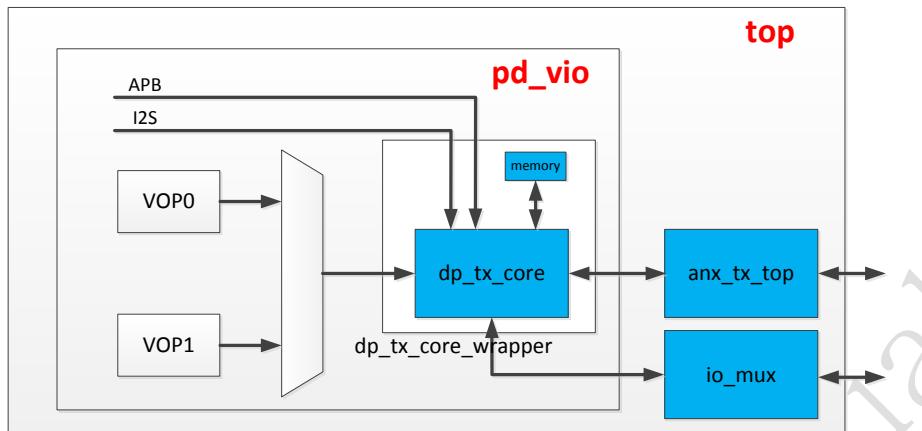


Fig. 34-2 eDP in SoC

There is a 512x8bits memory used to store the HDCP keys in the eDP controller. When GRF\_SOC\_CON8[13] (grf\_edp\_mem\_ctrl\_sel) = 1'b0, the memory is controlled by APB bus. When GRF\_SOC\_CON8[13] (grf\_edp\_mem\_ctrl\_sel) = 1'b1, the memory is controlled by eDP controller.

Please refer to “eDP TX IP Rockchip Datasheet.docx” for detail information.

## 34.4 Register Description

Please refer to “eDP TX IP Rockchip Datasheet.docx” for detail information.

## 34.5 Interface Description

### 34.5.1 Video Input Source

In RK3288, the eDP TX video source comes from vop0 or vop1.

- GRF\_SOC\_CON6[5] == 1'b0, video source from vop0.
- GRF\_SOC\_CON6[5] == 1'b1, video source from vop1.

### 34.5.2 Audio Input Source

In RK3288, the eDP TX audio source can come from I2S\_8CH and SPDIF, and the SPDIF source comes from SPDIF\_2CH or SPDIF\_8CH.

- GRF\_SOC\_CON2[1] == 1'b0, SPDIF source from SPDIF\_8CH.
- GRF\_SOC\_CON2[1] == 1'b1, SPDIF source from SPDIF\_2CH

### 34.5.3 Hot plug

There is a hot plug input signal to eDP TX. This signal is muxed with GPIO7\_B[3], and is enabled by “GPIO7B\_IOMUX[7:6] = 2'b10”.

## **34.6 Application Notes**

Please refer to "eDP TX IP Rockchip Datasheet.docx" for detail information.

Rockchip Confidential

## Chapter 35 MIPI-CSI PHY

### 35.1 Overview

The MIPI D-PHY is compliant with the MIPI D-PHY interface specification, revision 1.1. The D-PHY can be reused for both master and slave applications. The lane modules are bidirectional with HS-TX, HS-RX, LP-TX, LP-RX, and LP-CD functions.

The D-PHY is targeted for the digital data transmission between a host processor and display drivers or camera interfaces in mobile applications, supporting a maximum effective bit rate of 1.5Gbps per lane. The assembled four-data-lane system enables up to 6Gbps aggregate communication throughputs, delivering the bandwidth needed for high-throughput data transfer.

There were three D-PHY in RK3288, one is for DSI, one is for CSI, another can configure to DSI or CSI.

The MIPI D-PHY supports the following features:

- Attachable PLL clock multiplication unit for master-side functionality
- Flexible input clock reference – 5MHz to 500MHz
- 50% DDR output clock duty-cycle
- Lane operation ranging from 80Mbps to 1.5Gbps in forward direction
- Aggregate throughput up to 6Gbps with four data lanes
- PHY-Protocol Interface (PPI) for clock and data lanes
- Low-power Escape modes and Ultra Low Power state
- $1.8V \pm 10\%$  analog supply operation
- $1.0V \pm 10\%$  digital supply operation

### 35.2 Block Diagram

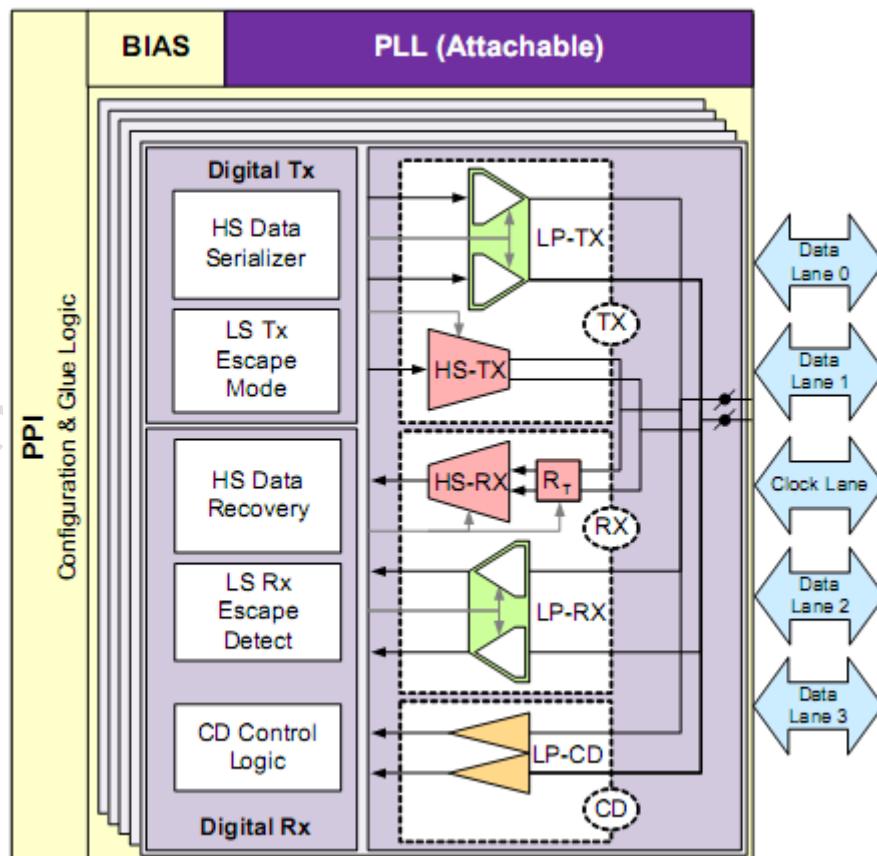


Fig. 35-1 MIPI D-PHY detailed block diagram

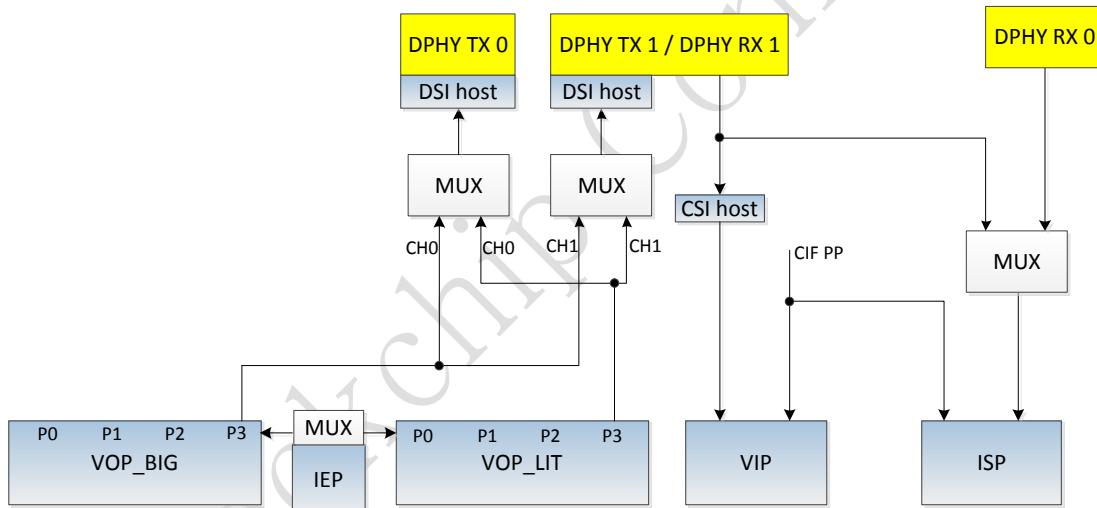
- HS Driver/Receiver
  - Implements high-speed TX and RX functionalities
  - Replicated for each lane
- LP Driver/Receiver
  - Implements the low-power TX and RX functionalities
  - Replicated for each lane
- Contention Detectors
  - Used for contention detector when there is a direction change in the low-power mode
  - Replicated for each lane
- PLL
  - Generates high-speed clocks required in Master Mode
  - Can be attached to D-PHY for Master applications
- Digital Block: Includes all control logic as well as PPI

MIPI CSI2 D-PHY configuration contains one Clock Lane Module and four Data Lane Modules. Each of these PHY Lane Modules communicates via two lines to a complementary part at the other side of the Lane Interconnect.

## 35.3 Function Description

### 35.3.1 System Connection

There are three D-PHY in RK3288, their connection are shown as following figure:



#### D-PHY RX0

D-PHY RX0 is only used for RX, receive the Mipi Camera data then send to ISP. In this mode, you must set grf\_con\_isp\_dphy\_sel (bit[1] of GRF\_SOC\_CON6) to 1'b0.

#### D-PHY TX0

D-PHY TX0 is only used for TX, send the data from VOP\_BIG or VOP\_LIT to the Mipi Panel. You can select data from VOP\_BIG or VOP\_LIT by setting grf\_con\_dsi0\_lcdc\_sel (bit[6] of GRF\_SOC\_CON6)

#### D-PHY TX1RX1

D-PHY TX1RX1 can configures to for TX or for RX.

The D-PHY can be configured to for TX by setting grf\_dphy\_tx1rx1\_masterslavez = 1'b0, and setting grf\_dphy\_tx1rx1\_basedir = 1'b0, and you can set the grf\_con\_dsi1\_lcdc\_sel (bit[9] of GRF\_SOC\_CON6) to select the data from VOP\_BIG or from VOP\_LIT.

If you want the D-PHY work as for RX, you must set grf\_dphy\_tx1rx1\_masterslavez = 1'b1, and set grf\_dphy\_tx1rx1\_basedir = 1'b1, then you must select the data from D-PHY RX1 to CSI Host or ISP by setting grf\_con\_isp\_dphy\_sel (bit[1] of GRF\_SOC\_CON6)

The detail register setting is as following table:

Table 35-1 Register Config For D-PHY Mode Select

TX0 + VOP_BIG	TX0 + VOP_LIT	TX1RX1 + VOP_BIG	TX1RX1 + VOP_LIT
bit[6] of GRF_SOC_CON6 = 1'b0	bit[6] of GRF_SOC_CON6 = 1'b1	bit[9] of GRF_SOC_CON6 = 1'b0	bit[9] of GRF_SOC_CON6 = 1'b1
		bit[14] of GRF_SOC_CON6 = 1'b0	bit[14] of GRF_SOC_CON6 = 1'b0
		bit[14] of GRF_SOC_CON14 = 1'b1	bit[14] of GRF_SOC_CON14 = 1'b1
		bit[15] of GRF_SOC_CON14 = 1'b0	bit[15] of GRF_SOC_CON14 = 1'b0
bit[8:7] of GRF_SOC_CON6	bit[8:7] of GRF_SOC_CON6	bit[11:10] of GRF_SOC_CON6	bit[11:10] of GRF_SOC_CON6
bit[11:0] of GRF_SOC_CON8	bit[11:0] of GRF_SOC_CON8	bit[15:0] of GRF_SOC_CON9	bit[15:0] of GRF_SOC_CON9
bit[10:8] of GRF_SOC_CON15	bit[10:8] of GRF_SOC_CON15	bit[12] of GRF_SOC_CON14	bit[12] of GRF_SOC_CON14
		bit[7:4] of GRF_SOC_CON15	bit[7:4] of GRF_SOC_CON15
bit[0] of GRF_SOC_CON16	bit[0] of GRF_SOC_CON16	bit[1] of GRF_SOC_CON16	bit[1] of GRF_SOC_CON16
RX0 + ISP	TX1RX1 + ISP	TX1RX1 + CSI_Host + VIP	
bit[1] of GRF_SOC_CON6 = 1'b0	bit[1] of GRF_SOC_CON6 = 1'b1		
	bit[14] of GRF_SOC_CON6 = 1'b1	bit[14] of GRF_SOC_CON6 = 1'b1	
	bit[13] of GRF_SOC_CON14 = 1'b1	bit[13] of GRF_SOC_CON14 = 1'b0	
	bit[14] of GRF_SOC_CON14 = 1'b0	bit[14] of GRF_SOC_CON14 = 1'b0	
	bit[15] of GRF_SOC_CON14 = 1'b1	bit[15] of GRF_SOC_CON14 = 1'b1	
bit[15:0] of GRF_SOC_CON10	bit[15:0] of GRF_SOC_CON9	bit[15:0] of GRF_SOC_CON9	
bit[10:0] of GRF_SOC_CON14	bit[12] of GRF_SOC_CON14	bit[12] of GRF_SOC_CON14	
bit[3:0] of GRF_SOC_CON15	bit[7:4] of GRF_SOC_CON15	bit[7:4] of GRF_SOC_CON15	

### 35.3.2 Operating Modes

This section describes the various operating modes of the MIPI D-PHY, the following Figure illustrates the various modes of the MIPI D-PHY during initialization and active operating mode.

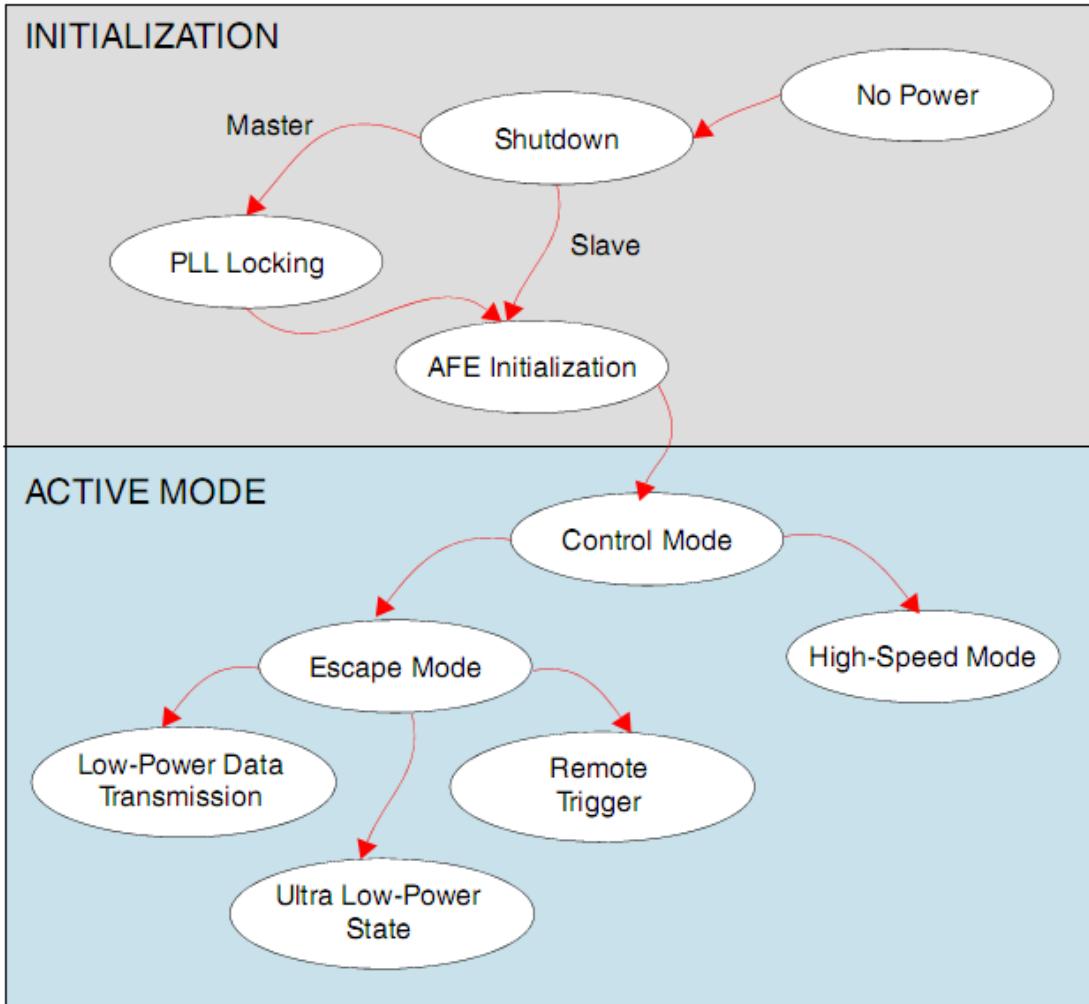


Fig. 35-2 MIPI D-PHY Initialization from Shutdown to Idle Modes

## Initialization

- **No Power Mode**

The No Power mode is characterized by the non-existence of any supply voltage applied to MIPI D-PHY. In order to get to the powered modes, proper voltages should be applied sequentially to MIPI D-PHY, this task is usually done by the SoC PMU or eventually by global powering up sequence.

The recommended powering up sequence is that the core voltage (VDD) powers up first and the I/O voltage (AVDD) powers up next. This is not considered as a constraint, but instead a guideline, as it results in the best-case operating scenario, where power-down currents are kept to a minimum.

For complex SoCs, it is likely that core voltage islands exist for different main blocks/macros, while the I/O voltage is always present. This means that VDD may power up after the I/O voltage and not follow the previous guideline. This is not considered a problem because MIPI D-PHY supports power collapsing, which ensures valid logical levels across power domains even when one of the supplies is not present at a given time.

The digital core voltage (VDD) and I/O analog voltage (AVDD) domains are isolated by the use of level shifter cells. No additional leakage is expected when there is a lack of VDD and/or AVDD.

- **Shutdown Mode**

This mode is the lowest power consumption mode, where all analog blocks are disabled, and digital logic is reset. The current consumption is given by the analog stand-by current and the

digital logic leakage current. It is entered asynchronously when RSTZ and SHUTDOWNZ are in low state. It should be ensured that the TESTCLR signal is asserted by default, as it acts as an active high reset to the control block responsible for the configuration values preset.

In this mode, the differential lines of DATAN/DATAP and CLKN/CLKP are high impedance (Hi-Z).

Depending on the MIPI D-PHY usage, some additional steps can be performed. By default, MIPI D-PHY is configured to work only on the lower operation range of 80-110 Mbps. If higher bit rate operation is required, you should set the register hsfreqrange (HS RX Control of Lane 0) with the proper code. If MIPI D-PHY is expected to work always at the same bit rate, this additional step can be performed while in Shutdown mode as the control interface is independent of the rest of MIPI D-PHY. Conversely, if the MIPI D-PHY is expected to change the bit rate after initialization, hsfreqrange should be updated while in Idle mode. For more information on these options, see "Active Modes".

In addition, when working in Master mode, the PLL must be configured to select the proper input frequency and the desired output frequency, which determines the bit rate on the transmission path. For more information, see "PLL Requirements".

When RSTZ and SHUTDOWNZ are set to logic high level, MIPI D-PHY leaves this state and starts an initialization procedure.

- **PLL Locking Mode and AFE Initialization**

The MIPI D-PHY consists of four data lanes, but applications can use four or lesser number data lanes. In such cases, you are granted access to individual enabling signals (ENABLE\_N) that control which lanes should be used and evolve through all the necessary initialization steps. It is assumed that such configurations are static or at least are stable prior to leaving the Shutdown mode.

After the reset signals (RSTZ and SHUTDOWNZ) are released, the MIPI D-PHY begins an initialization sequence that allows its correct operation. Sequence of the release of signals is not critical but it is recommended that SHUTDOWNZ precedes RSTZ. It is also assumed that the CFG\_CLK signal is available and stable by that time.

If there are no test or configuration operations to be performed, the TESTCLR signal can be kept at logic high level. Otherwise, the TESTCLR must be de-asserted to bring the control logic out of reset and allow for the necessary configuration steps through the control interface.

The D-PHY specification has many timing intervals which have to be followed to ensure proper operation. The fact that those timing intervals often have both a relative Unit Interval (UI) and absolute timing components, makes it difficult to meet the maximum and minimum values across the complete data rate range (80 Mbps-1.5 Gbps) by just using default settings. To cope with this, MIPI D-PHY implements a set of frequency ranges that needs to be configured prior to starting normal operation. Those ranges, when in Master operation, also define the operating bit rate, assuming REFCLK is equal to 27MHz. If the desired bit rate or REFCLK frequencies are different, directly configure the PLL as described in "PLL Requirements". All these steps come under the category of configurations that need to be performed through the control interface with TESTCLR de-asserted.

The following Table lists the frequency ranges.

Table 35-2 Frequency Ranges

<b>Range (Mbps)</b>	<b>hsfreqrange[5:0]</b>	<b>Default Bit Rate (Mbps)</b>
80-90 (default)	000000	81
90-100	010000	90
100-110	100000	108

110-130	000001	126
130-140	010001	135
140-150	100001	144
150-170	000010	162
170-180	010010	180
180-200	100010	198
200-220	000011	216
220-240	010011	234
240-250	100011	243
250-270	000100	270
270-300	010100	297
300-330	000101	324
330-360	010101	360
360-400	100101	396
400-450	000110	450
450-500	010110	486
500-550	000111	540
550-600	010111	594
600-650	001000	648
650-700	011000	684
700-750	001001	738
750-800	011001	783
800-850	101001	846
850-900	111001	900
900-950	001010	945
950-1000	011010	999
1000-1050	101010	1044
1050-1100	111010	1080
1100-1150	001011	1134
1150-1200	011011	1188
1200-1250	101011	1242
1250-1300	111011	1296

1300-1350	001100	1350
1350-1400	011100	1386
1400-1450	101100	1440
1450-1500	111100	1494

The hsfreqrange field is accessible through control code 0x44 ("HS RX Control of Lane 0") when TESTDIN[7] = 0 and TESTDIN[0] = 0. The hsfreqrange[5:0] field is programmed with the contents of TESTDIN[6:1] at every rising edge of TESTCLK.

If the MIPI D-PHY is configured to work as a Master (MASTERSLAVEZ=1'b1), the PLL becomes active and MIPI D-PHY goes through the PLL Locking mode, in which the MIPI D-PHY waits for the PLL to acquire lock, indicated by the LOCK output going high. A valid REFCLK (FREFCLK) should be provided.

Following the PLL lock, the rest of the AFE is initialized leading to the enabling of the low-power drivers. After completing these transitory states, the lines go to the Stop state (LP = 11) and the TX achieves active mode.

In the case of a Slave configuration (MASTERSLAVEZ = 1'b0), PLL is inactive, therefore only the rest of AFE initialization takes place.

The initialization sequence determines that bandgap and biasing blocks are enabled first. After the related voltage and current references get settled, a second step is triggered where the internal calibrations are performed, and this can include internal resistors, receivers offset compensation, and so on.

When this second step is complete, the control is passed to the lanes, which handle the power management for LP/HS requests, enabling or disabling the corresponding drivers and receivers.

All initialization steps are performed once the STOPSTATEDATA\_N and STOPSTATECLK outputs get asserted.

Initialization period (TINIT) is a protocol dependent parameter with a minimum 100  $\mu$ s defined by the specification. The MIPI D-PHY does not set any limit to the initialization period, meaning it drives a Stop state (LP-11) immediately after the AFE initialization and PLL lock when in Master mode, or alternatively starts decoding the LP commands after the AFE initialization in Slave mode. It is up to the controller or the upper layers to ensure the proper initialization times through the correct handling of MIPI D-PHY control signals. This time must conform to D-PHY specification and obey the minimum specified 100  $\mu$ s value.

Following Figure shows a possible power-up sequence for a Slave application when the default setting is 80-110 Mbps operation.

If the desired operation mode is different from the default one, additional configuration steps can be performed during the T2+T3 time window.

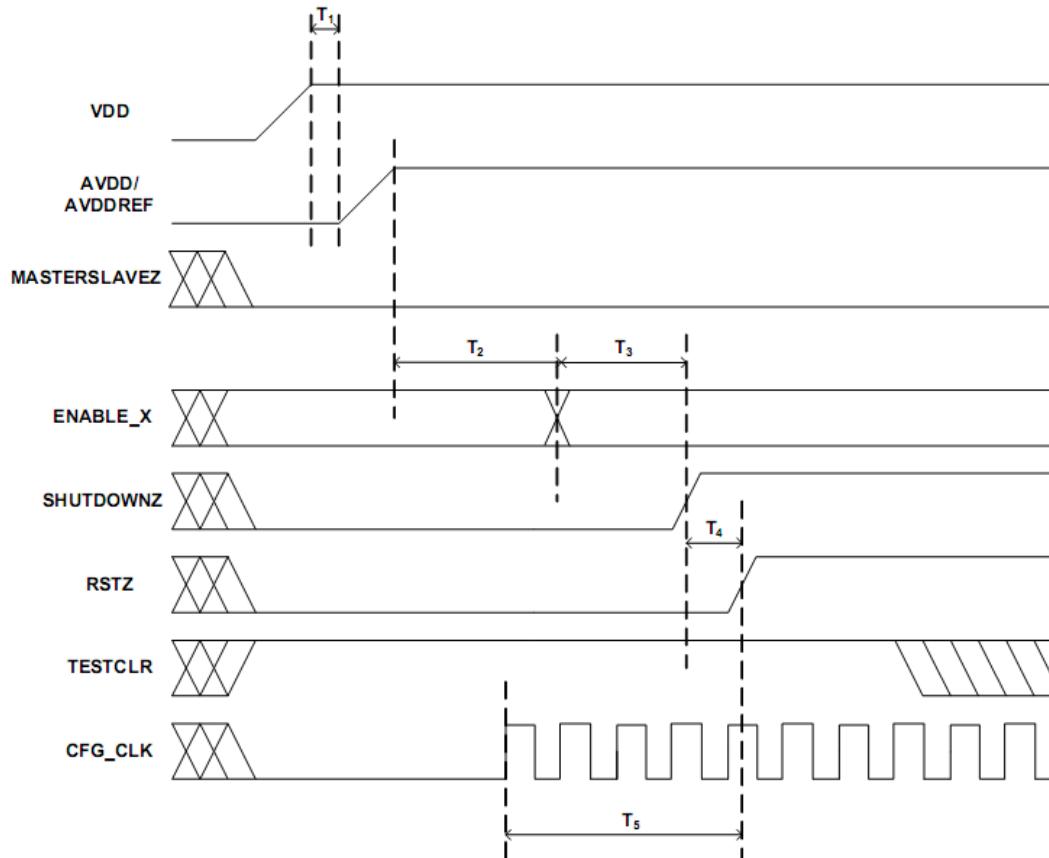


Fig. 35-3 Power-Up Sequence for Slave Operation

Table 35-3 Power-Up Sequence Timings

Parameter	Symbol	Minimum	Typical	Units
Delay from stable VDD to AVDD/AVDDREF start	T1	0	1	us
Delay from stable AVDD/AVDDREF to ENABLE_X definition	T2	0		ns
Delay for assertion of SHUTDOWNZ after ENABLE_X definition	T3	5		ns
Delay from SHUTDOWNZ assertion to RSTZ assertion	T4	5		ns
Time for CFG_CLK setting before the assertion of RSTZ	T5	1		CFG_CLK

## Active Modes

- Idle Mode

Idle mode is the default operating mode. After the initialization is completed (analog calibrations and PLL locking for Master configurations), the MIPI D-PHY remains in this default mode until some request is placed. The request is placed either by the protocol layer for TX, or directly through the sequence of low-power signals in the lanes in case of RX. While in control mode, the transmitter side sets the LP-11 state in the lines - this is called the Stop state. The receiver side remains in control mode while receiving LP-11 in the lines. Any request must start from and end in Stop state. Following a request, a lane can leave control mode for either high-speed data transfer mode, Escape mode, or Ultra Low Power state.

- **High-Speed Data Transfer Mode**

Once the initialization sequence is completed, the MIPI D-PHY remains in control mode, which is the default operating mode, until some request appears. High-speed is one of the possible requests at this point. High-speed data transfer occurs in bursts. Only during these bursts the lane is in high-speed mode. A high-speed burst must start from and return to a Stop state (control mode). A high-speed burst allows for the transmission of payload data by the data lanes. Inherent to such data transmission is the existence of a valid DDR clock in the clock lane.

High-speed data bursts are independent for each lane, which means that each data lane can start and end a high-speed transmission independently of the state of the remaining data lanes.

A burst contains the low-power initialization sequence, the high-speed data payload, and also the end of transmission sequence.

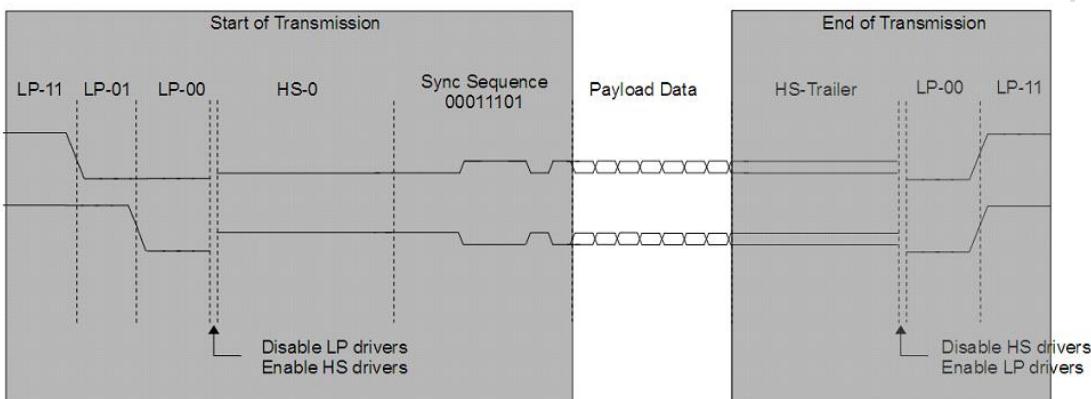


Fig. 35-4 HS Data Transfer Sequence

From the transmitter side, high-speed mode is entered when the corresponding TXREQUESTHS input is set high (assuming that MIPI D-PHY is in Stop state). This request is processed in a slightly different way for clock and data lanes. For a clock lane, the high-speed request is followed by the transmission of a low-power sequence that represents this request for the receiver side (a lane high-speed request). Only after generating this sequence the low-power driver is disabled, and the high-speed driver enabled. After the time necessary to settle, the transmission of the high-speed DDR clock starts. For a data lane, the high-speed request also starts with a lane high-speed request, and in addition, extend the payload data with a leader and a trailer sequence that allow for the receiver synchronization. The transmission of such sequence requires the existence of a valid high-speed clock signal in the clock lane.

When the high-speed request input is disabled, each lane leaves the high-speed data transmission mode. It is important that a clock lane must be in high-speed mode during the complete high-speed data transmission state of all the lanes. The clock lane must enter the high-speed mode before a high-speed data transmission begins and it must not leave this state before all the lanes finish their respective high-speed data transmission bursts. The operation sequence when leaving the high-speed mode is also slightly different for data and clock lanes. For a clock lane, the high-speed transmission always ends with a HS-0 state, followed by the disabling of the high-speed driver, and enabling of low-power driver. As for a data lane, the transmission ends with the differential state opposite to the last bit transmitted, followed by the disabling of the high-speed driver, and enabling of the low-power driver.

The receiver side enters the high-speed mode following the sequence of low-power states in the lines: LP-11, LP-01, and LP-00. This sequence is seen as a high-speed mode request, and toggles the enabling of the high-speed receivers. The synchronization is then achieved through the identification of the leader sequence in the received differential high-speed data. Once the synchronization is achieved, the MIPI D-PHY outputs the received bytes through the protocol layer, until a Stop state (LP-11) is detected in the lane.

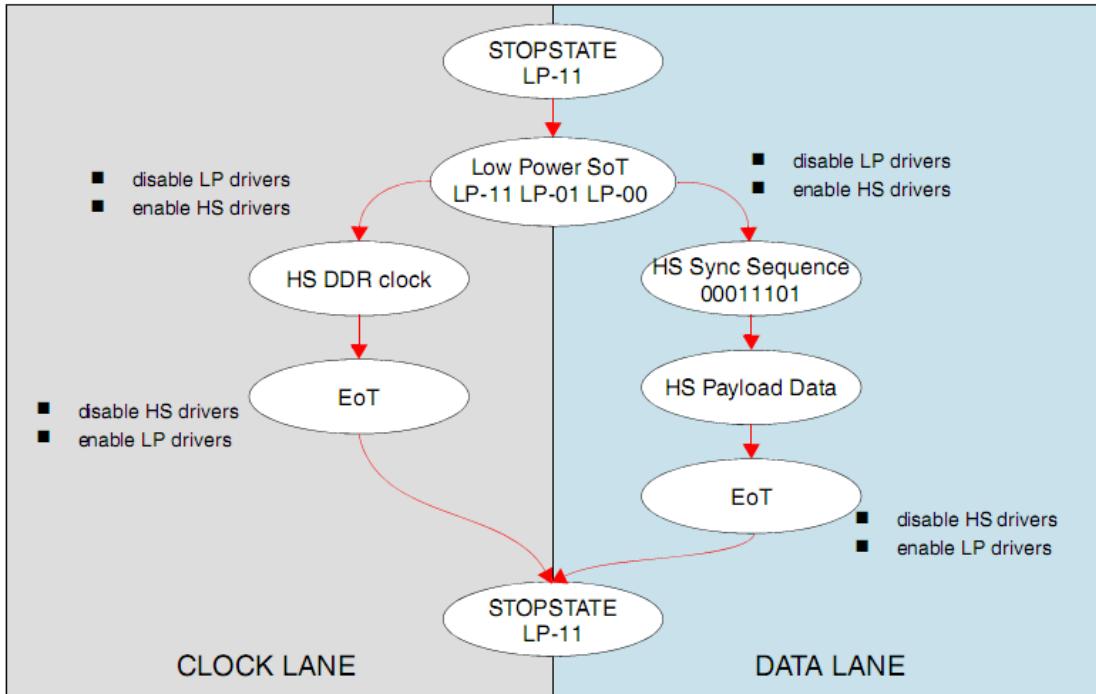


Fig. 35-5 HS Data Transfer State Diagram

The current implementation does not feature EoT processing which should be done at the controller level. This affects the behavior of the RXActiveHs and RXValidHS signals as illustrated in "Timing Diagrams".

#### ● Escape Mode

Escape mode is a special mode of operation that uses the data lanes to communicate asynchronously using the low-power states at low-speed. The MIPI D-PHY supports this mode in both directions. A Data Lane enters the Escape mode through an Escape mode entry procedure (LP-11, LP-10, LP-00, LP-01, LP-00), if an LP-11 is detected before reaching LP-00 state, the entry is aborted and the receiver returns to the Stop state. Once the sequence is correctly completed, the transmitter sends an 8-bit command to indicate a requested action. The following Table shows the Escape mode supported actions. If the entry command is not valid, it is ignored, ERRESC error flag goes high, and the receiver waits until the transmitter returns to the Stop state. The MIPI D-PHY applies Spaced-One-Hot encoding (a Mark state is interleaved with a Space state) on commands and data.

Each symbol consists of the following two parts:

- One-Hot phase
- Space state

To transmit one bit, a Mark-1 should be sent followed by the Space state. In the case of a zero bit, a Mark-0 should be sent followed by Space state.

Table 35-4 Possible Escape Mode Sequences for Data Lanes

Escape Mode Action	Entry Command Pattern	Command Type
Low-Power Data Transmission	8'b11100001	mode
Ultra-Lower Power State	8'b00011110	mode
Reset Trigger	8'b01100010	trigger
Unknown-3	8'b01011101	trigger

Unknown-4	8'b00100001	trigger
Unknown-5	8'b10100000	trigger

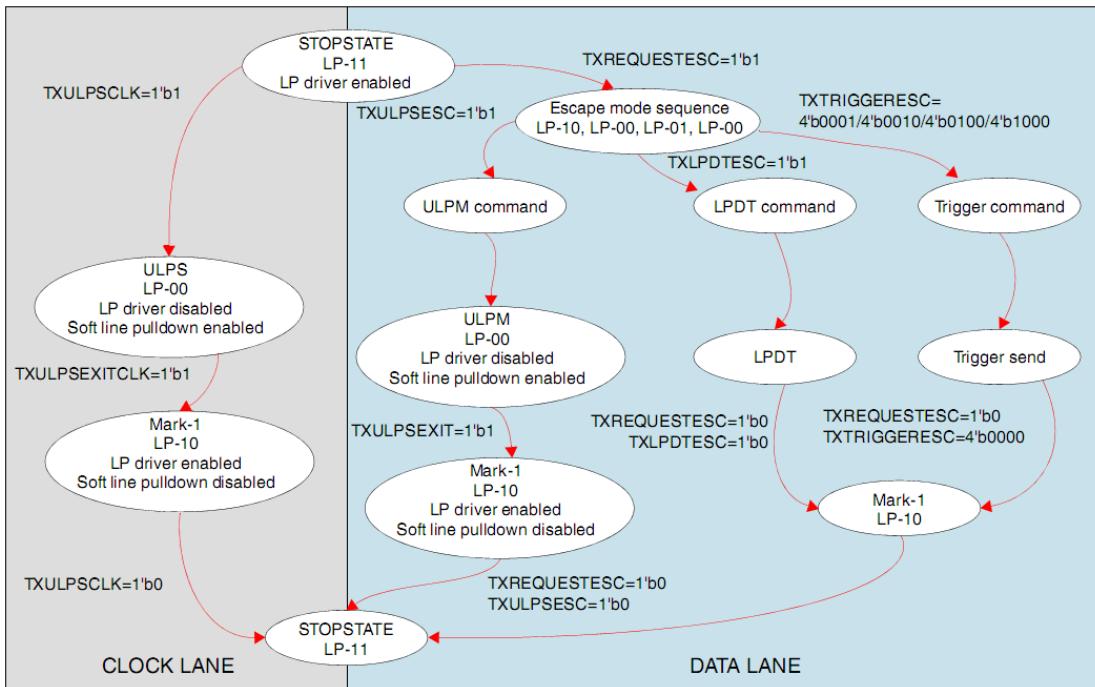


Fig. 35-6 Escape Mode Sequences State Diagram

## 35.4 Register Description

### 35.4.1 Registers Summary

Ref 9.6 Control/Test Codes of MIPI D-PHY Bidir 4L for GF28-nm SLP/1.8V Databook

### 35.4.2 Detail Register Description

Ref 9.6 Control/Test Codes of MIPI D-PHY Bidir 4L for GF28-nm SLP/1.8V Databook

## 35.5 Application Notes

### 35.5.1 PLL Requirements

Ref 6 PLL Requirements of MIPI D-PHY Bidir 4L for GF28-nm SLP/1.8V Databook

### 35.5.2 Calibration Requirements

Ref 7 Calibration Requirements of MIPI D-PHY Bidir 4L for GF28-nm SLP/1.8V Databook

### 35.5.3 Electrical and Timing Information

The following Tables provides the electrical and timing characteristics of MIPI D-PHY, the following conditions are applicable unless otherwise noted:

- Vdd (core) = 1.0V
- Vdd (I/O) = 1.8V
- Ta<sup>o</sup> = T<sub>min</sub> to T<sub>max</sub>

Table 35-5 DC Specifications

Symbol	Parameter	Condition	Min	Typ	Max	Unit
<b>Input DC Specifications – Apply to CLKP/N and DATAP/N Inputs</b>						
V <sub>I</sub>	Input signal voltage range		-50		1350	mV
I <sub>LEAK</sub>	Input leakage current	V <sub>GNDH(min)</sub> ≤ V <sub>I</sub> ≤ V <sub>GNDH(max)</sub> + V <sub>OH(absmax)</sub> Lane module in LP receive mode				uA
V <sub>GNDH</sub>	Ground shift		-50		50	mV
V <sub>OH(absmax)</sub>	Maximum transient output voltage level		-0.15		1.45	V
t <sub>VOH(absmax)</sub>	Maximum transient time above V <sub>OH(absmax)</sub>				20	ns
<b>HS Line Drivers DC Specifications</b>						
V <sub>OD</sub>	HS Transmit Differential output voltage magnitude	80 Ω ≤ R <sub>L</sub> ≤ 125 Ω	140	200	270	mV
Δ V <sub>OD</sub>	Change in Differential output voltage magnitude between logic states	80 Ω ≤ R <sub>L</sub> ≤ 125 Ω			14	mV
V <sub>CMTX</sub>	Steady-state common-mode output voltage	80 Ω ≤ R <sub>L</sub> ≤ 125 Ω	150	200	250	mV
ΔV <sub>CMTX(1,0)</sub>	Changes in steady-state common-mode output voltage between logic states	80 Ω ≤ R <sub>L</sub> ≤ 125 Ω			5	mV
V <sub>OHHS</sub>	HS output high voltage	80 Ω ≤ R <sub>L</sub> ≤ 125 Ω			360	mV
Z <sub>os</sub>	Single-ended output impedance		40	50	62.5	Ω
ΔZ <sub>os</sub>	Single-ended output impedance mismatch				10	%
<b>LP Line Drivers DC Specifications</b>						
V <sub>OL</sub>	Ouput Low-level SE voltage		-50		50	mV
V <sub>OH</sub>	Ouput high-level SE voltage		1.1	1.2	1.3	V
Z <sub>OLP</sub>	Single-ended output impedance		110			Ω
ΔZ <sub>OLP(01,10)</sub>	Single-ended ouput impedance mismatch driving opposite level				20	%
ΔZ <sub>OLP(00,11)</sub>	Single-ended ouput impedance mismatch driving same level				5	%
<b>HS Line Receiver DC Specifications</b>						

$V_{IDTH}$	Differential input high voltage threshold			70	mV
$V_{IDTL}$	Differential input low voltage threshold		-70		mV
$V_{IHHS}$	Single ended input high voltage			460	mV
$V_{ILHS}$	Single ended input low voltage		-40		mV
$V_{CMRXDC}$	Input common mode voltage		70	330	mV
$Z_{ID}$	Differential input impedance		80	125	$\Omega$

**LP Line Receiver DC Specifications**

$V_{IL}$	Input low voltage			550	mV
$V_{IH}$	Input high voltage		880		mV
$V_{HYST}$	Input hysteresis		25		mV

**Contention Line Receiver DC Specifications**

$V_{ILF}$	Input low fault threshold			200	mV
$V_{IHF}$	Input high fault threshold		450		mV

**35.5.4 Switching Characteristics**

This section provides the various specifications for the switching characteristics of MIPI D-PHY

Table 35-6 Switching Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Unit
<b>Configuration Clock Specifications</b>						
$F_{CFG\_CLK}$	CFG_CLK frequency		17		27	MHz
$DC_{CFG\_CLK}$	CFG_CLK duty cycle		40	50	60	%
<b>HS Line Drivers AC Specifications</b>						
-	Maximum Serial Data rate (forward direction)	On DATAP/N outputs. $80 \Omega \leq R_L \leq 125 \Omega$	80		1500	Mbps
$F_{DDRCLK}$	DDR CLK frequency	On CLKP/N outputs	40		750	MHz
$P_{DDRCLK}$	DDR CLK period	$80 \Omega \leq R_L \leq 125 \Omega$	1.3		25	ns
$UI_{INST}$	UI instantaneous				12.5	ns <sup>a</sup>
$\Delta UI$	UI variation		-10%		10%	UI <sup>b</sup>

			-5%		5%	UI <sup>c</sup>
t <sub>CDC</sub>	DDR CLK duty cycle	t <sub>CDC</sub> = t <sub>CPH</sub> /P <sub>DDRCLK</sub>		50		%
t <sub>CPH</sub>	DDR CLK high time			1		UI
t <sub>CPL</sub>	DDR CLK low time			1		UI
-	DDR CLK/DATA Jittler <sup>d</sup>			75		ps pk-pk
t <sub>SKEW[PN]</sub>	Intra-Pair (Pulse) skew			0.075		UI
t <sub>SKEW[TX]</sub>	Data to Clock Skew		-0.15		0.15	UI <sup>e</sup>
			-0.20		0.20	UI <sup>f</sup>
t <sub>SETUP[RX]</sub>	Data to Clock Receiver Setup time		0.15			UI <sup>g</sup>
			0.20			UI <sup>h</sup>
t <sub>HOLD[RX]</sub>	Clock to Data Receiver Hold time		0.15			UI <sup>g</sup>
			0.20			UI <sup>h</sup>
t <sub>r</sub>	Differential output signal rise time	20% to 80%, R <sub>L</sub> =50Ω			0.30	UI <sup>i</sup>
					0.35	UI <sup>j</sup>
			100			ps <sup>k</sup>
t <sub>f</sub>	Differential output signal fall time	20% to 80%, R <sub>L</sub> =50Ω			0.30	UI <sup>i</sup>
					0.35	UI <sup>j</sup>
			100			ps <sup>k</sup>
ΔV <sub>CMTX(HF)</sub>	Common level variation above 450MHz	80 Ω ≤ R <sub>L</sub> ≤ 125 Ω			15	mVrms
ΔV <sub>CMTX(LF)</sub>	Common level variation between 50MHz and 450MHz	80 Ω ≤ R <sub>L</sub> ≤ 125 Ω			25	mVp

Table 35-7 AC Specifications

Symbol	Parameter	Condition	Min	Typ	Max	Unit
<b>LP Line Drivers AC Specifications</b>						
t <sub>rlp</sub> , t <sub>flp</sub>	Single ended output rise/fall time	15% to 85%, C <sub>L</sub> <70pF			25	ns
t <sub>reot</sub>		30% to 85%, C <sub>L</sub> <70pF			35	ns
∂V/∂t <sub>SR</sub>	Signal slew rate <sup>l</sup>	15% to 85%, C <sub>L</sub> <70pF			150	mV/ns

$C_L$	Load capacitance		0		70	pF <sup>m</sup>
<b>HS Line Receiver AC Specifications</b>						
$\Delta V_{CMRX(HF)}$	Common mode interference beyond 450MHz				200	mVpp
$\Delta V_{CMRX(LF)}$	Common mode interference between 50MHz and 450MHz		-50		50	mVpp
$C_{CM}$	Common mode termination				60	pF <sup>n</sup>
<b>LP Line Receiver AC Specifications</b>						
$e^{SPIKE}$	Input pulse rejection				300	V.ps
$T_{MIN}$	Minimum pulse response		20			ns
$V_{INT}$	Pk-to-Pk interference voltage				300	mVpp
$f_{INT}$	Interference frequency		450			MHz
<b>Model Parameters Used for Driver Load Switching Performance Evaluation</b>						
$C_{PAD}$	Equivalent Single ended I/O PAD capacitance				1	pF
$C_{PIN}$	Equivalent Signal Ended Package + PCB capacitance				2	pF
$L_S$	Equivalent wire bond series inductance				1.5	nH
$R_S$	Equivalent wire bond series resistance				0.15	$\Omega$
$R_L$	Load Resistance		80	100	125	$\Omega$

**Notes:**

- a. This value corresponds to a minimum Mbps data rate.
- b. When  $UI \geq 1\text{ns}$ , within a single burst.
- c. When  $UI < 1\text{ns}$ , within a single burst.
- d. Jitter specification with clean clock at REFCLK input.
- e. Total silicon and package skew delay budget of  $0.3 * UIINST$  when D-PHY is supporting maximum data rate = 1 Gbps.
- f. Total silicon and package skew delay budget of  $0.4 * UIINST$  when D-PHY is supporting maximum data rate > 1 Gbps.
- g. Total setup and hold window for receiver of  $0.3 * UIINST$  when D-PHY is supporting maximum data rate = 1 Gbps.
- h. Total setup and hold window for receiver of  $0.4 * UIINST$  when D-PHY is supporting maximum data rate > 1 Gbps.
- i. Applicable when operating at HS bit rates  $\leq 1\text{ Gbps}$  ( $UI \geq 1\text{ ns}$ ).
- j. Applicable when operating at HS bit rates > 1 Gbps ( $UI < 1\text{ ns}$ ).

*k. Applicable for all HS bit rates. However, to avoid excessive radiation, bit rates  $\leq 1 \text{ Gbps}$  ( $UI \geq 1 \text{ ns}$ ), should not use values below 150 ps.*

*l. Measured as average across any 50 mV of the output signal transition.*

*m. CLOAD includes the low-frequency equivalent transmission line capacitance. The capacitance of TX and RX are assumed to always be  $< 10 \text{ pF}$ . The distributed line capacitance can be up to 50pF for a transmission line with 2ns delay.*

*n. For higher bit rates a 14pF capacitor will be needed to meet the common-mode return loss specification.*

## Chapter 36 MIPI CSI-2 Host Controller

### 36.1 Overview

The CSI-2 Host Controller implements the CSI-2 protocol on the host side. The CSI-2 link protocol specification is a part of communication protocols defined by MIPI Alliance standards intended for mobile system chip-to-chip communications. The CSI-2 specification is for the image application processor communication in cameras.

The CSI-2 Host Controller is designed to receive data from a CSI-2 compliant camera sensor. A D-PHY configured as a Slave acts as the physical layer.

The MIPI CSI-2 Host Controller supports the following features:

- Compliant with MIPI Alliance Specification for CSI-2, Version 1.01.00-9 November 2010
- Interface with MIPI D-PHY following PHY Protocol Interface, as defined in MIPI Alliance Specification for D-PHY, Version 1.1-7 November 2011
- Up to four D-PHY RX data lanes
- Dynamically configurable multi-lane merging
- Long and Short packet decoding
- Timing accurate signaling of Frame and Line synchronization packets
- Several Frame formats
  - General Frame or Digital Interlaced Video with or without accurate sync timing
  - Data Type (Packet or Frame Level) and Virtual Channel interleaving
- 32-bit Image Data Interface delivering data formatted as recommended in CSI-2 Specification
- All primary and secondary data formats
  - RGB, YUV, and RAW color space definitions
  - From 24-bit down to 6-bit per pixel
  - Generic or user-defined byte-based data types
- Error detection and correction
  - PHY level
  - Packet level
  - Line level
  - Frame level

### 36.2 Block Diagram

The following diagram shows the MIPI CSI-2 Host Controller architecture.

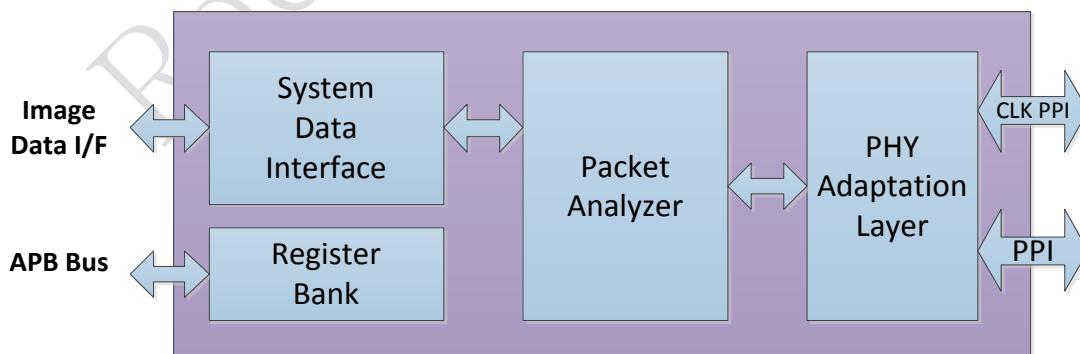


Fig. 36-1 MIPI CSI-2 Host Controller architecture

**PHY Adaptation Layer:** Manages the D-PHY PPI interface

**Packet Analyzer:** Merges the data from the different lanes

**Image Data Interface:** Reorders pixels into 32-bit data for memory storage and generates

timing accurate video synchronization signals

**AMBA-APB Register Bank:** Provides access to configuration and control registers

## 36.3 Function Description

### 36.3.1 Supported Resolutions and Frame Rates

The CSI-2 specification does not define the supported standard resolutions or frame rates. Camera sensor resolution, blanking periods, synchronization events, frame rates, and pixel color depth play a fundamental role in the required bandwidth. All these variables make it difficult to define a standard procedure to estimate the minimum lane rate and the minimum number of lanes that support a specific CSI-2 device.

Table 37-1 presents some predefined and supported camera settings, assuming the following:

- Clock lane frequency is 500 MHz or 750 MHz that results in a bandwidth of 1 Gbps or 1.5 Gbps respectively, for each data lane.
- No significant control/reserved traffic is present on the link when pixel data is being transmitted.

The last column of Table 37-1 presents the minimum number of lanes required for each configuration.

Table 36-1 Supported Camera Settings

Mega Pixels	Mega Pixels with Overhead	Refresh Rate (Hz)	Color Depth (bpp)	CSI2 BW (Mbits)	D-PHY at 1 Gbps Number of Lanes	D-PHY at 1.5Gbps Number of Lanes
2MP	2560000	15	24	922	1	1
2MP	2560000	30	24	1843	2	2
3MP	3840000	15	16	922	1	1
3MP	3840000	30	16	1843	2	2
3MP	3840000	30	24	2765	3	2
5MP	6400000	15	16	1536	2	2
5MP	6400000	15	24	2304	3	2
5MP	6400000	30	16	3072	4	3
8MP	10240000	15	16	2458	3	2
8MP	10240000	15	24	3686	4	3
8MP	10240000	30	12	3686	4	3
12MP	15360000	15	12	2765	3	2
12MP	15360000	15	16	3686	4	3
14MP	17920000	15	12	3226	4	3
16MP	20480000	15	12	3686	4	3
<b>Video Formats</b>						
1280x720 pixels(720p)	921600	30	24	664	1	1
1280x720 pixels(720p)	921600	60	24	1327	2	1
1920x1080 pixels(1080 p)	2073600	60	24	2986	3	2

### 36.3.2 Error Detection

The CSI-2 Host Controller analyzes the received packets and determines if there are protocol errors. It is possible to monitor the following errors:

- Frame errors such as incorrect Frame sequence, reception of a CRC error in the most recent frame, and the mismatch between Frame Start and Frame End
- Line errors such as incorrect line sequence and mismatch between Line Start and Line End
- Packet errors such as ECC or CRC mismatch
- D-PHY errors such as synchronization pattern mismatch

Table 37-2 shows all the errors that CSI-2 Host Controller can identify.

Table 36-2 Errors Identified by the CSI-2 Host Controller

Error	Description	Level	Action
-------	-------------	-------	--------

phy_errsotsynchs_*	Start of transmission error on data lane* with no synchronization achieved	PHY	Packets with this error are not delivered in IDI interface
phy_erresc_*	Escape entry error (ULPM) on data lane*	PHY	Informative only. Error is acknowledged in the register and the interrupt pin is raised.
phy_errsoths_*	Start of transmission error on data lane* but synchronization can still be achieved	PHY	Informative only since PHY can recover from this error. Error is acknowledged in register and the interrupt pin is raised.
vc*_err_crc	Checksum error detected on virtual channel*	Packet	Informative only. Error is acknowledged in the register and Interrupt pin is raised.
vc*_err_crc	Header ECC contains one error detected on virtual channel*	Packet	Informative only since controller can recover the correct header. Error is acknowledged in the register and the interrupt pin is raised.
err_ecc_double	Header ECC contains two errors. Unrecoverable.	Packet	Packets with this error are not delivered in IDI.s
err_id_vc*	Unrecognized or unimplemented data type detected in virtual channel*	Packet	Informative only. Error is acknowledged in the register and the interrupt pin is raised
err_f_bndry_match_vc*	Error matching Frame Start with Frame End for virtual channel*	Frame	Informative only. Error is acknowledged in register and the interrupt pin is raised if not masked.
err_f_seq_vc*	Incorrect Frame Sequence detected in virtual channel*	Frame	Informative only. Error is acknowledged in register and the interrupt pin is raised if not masked.
err_frame_data_vc*	Last received frame, in virtual channel*, had at least one CRC error	Frame	Informative only. Error is acknowledged in the register and the interrupt pin is raised.

## 36.4 Register Description

This section describes the control/status registers of the design.

### 36.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
CSIHOST_VERSION	0x0000	W	0x00000000	Version of the CSI2 Host
CSIHOST_N_LANES	0x0004	W	0x00000001	Number of active data lanes
CSIHOST_PHY_SHUT_DOWNZ	0x0008	W	0x00000000	PHY shutdown control
CSIHOST_DPHY_RST_Z	0x000c	W	0x00000000	DPHY reset control
CSIHOST_CSI2_RESETN	0x0010	W	0x00000000	CSI-2 Controller reset
CSIHOST_PHY_STAT_E	0x0014	W	0x00000000	General settings for all blocks
CSIHOST_ERR1	0x0020	W	0x00000000	Error state register 1
CSIHOST_ERR2	0x0024	W	0x00000000	Error state register 2
CSIHOST_MSK1	0x0028	W	0x00000000	Masks for errors 1
CSIHOST_MSK2	0x002c	W	0x00000000	Masks for errors 2
CSIHOST_PHY_TEST_CTRL0	0x0030	W	0x00000000	D-PHY test interface control 0

Name	Offset	Size	Reset Value	Description
CSIHOST_PHY_TEST_CTRL1	0x0034	W	0x00000000	D-PHY test interface control 1

Notes: Size: **B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**-WORD (32 bits) access

### 36.4.2 Detail Register Description

#### CSIHOST\_VERSION

Address: Operational Base + offset (0x0000)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	VERSION Version of the mipi csi2 host

#### CSIHOST\_N\_LANES

Address: Operational Base + offset (0x0004)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1:0	RW	0x1	N_LANES Number of active data lanes 00: 1 data lane (lane 0) 01: 2 data lanes (lanes 0 and 1) 10: 3 data lanes (lanes 0, 1, and 2) 11: 4 data lanes (All) Can only be updated when the D-PHY lane is in Stop state.

#### CSIHOST\_PHY\_SHUTDOWNZ

Address: Operational Base + offset (0x0008)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	PHY_SHUTDOWNZ D-PHY shutdown input. active low

#### CSIHOST\_DPHY\_RSTZ

Address: Operational Base + offset (0x000c)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	DPHY_RSTZ D-PHY reset output. active low

#### CSIHOST\_CSII\_RESETN

Address: Operational Base + offset (0x0010)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	CSI2_RESETN CSI-2 controller reset output. active low

#### CSIHOST\_PHY\_STATE

Address: Operational Base + offset (0x0014)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:11	RO	0x0	reserved
10	RO	0x0	PHY_STOPSTATECLK Clock lane in stop state
9	RO	0x0	PHY_RXULPSCLKNOT This signal indicates that the clock lane module has entered the Ultra Low Power state, active low
8	RO	0x0	PHY_RXCLKACTIVEHS Indicates that the clock lane is actively receiving a DDR clock
7	RO	0x0	PHY_STOPSTATEDATA_3 Data lane 3 in stop state
6	RO	0x0	PHY_STOPSTATEDATA_2 Data lane 2 in stop state
5	RO	0x0	PHY_STOPSTATEDATA_1 Data lane 1 in stop state
4	RO	0x0	PHY_STOPSTATEDATA_0 Data lane 0 in stop state
3	RO	0x0	PHY_RXULPSESC_3 lane module 3 has entered the Ultra Low Power mode
2	RO	0x0	PHY_RXULPSESC_2 lane module 2 has entered the Ultra Low Power mode
1	RO	0x0	PHY_RXULPSESC_1 lane module 1 has entered the Ultra Low Power mode
0	RO	0x0	PHY_RXULPSESC_0 lane module 0 has entered the Ultra Low Power mode

**CSIHOST\_ERR1**

Address: Operational Base + offset (0x0020)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28	RO	0x0	err_ecc_double Header ECC contains 2 errors, unrecoverable
27	RO	0x0	vc3_err_crc Checksum error detected on virtual channel 3
26	RO	0x0	vc2_err_crc Checksum error detected on virtual channel 2
25	RO	0x0	vc1_err_crc Checksum error detected on virtual channel 1
24	RO	0x0	vc0_err_crc Checksum error detected on virtual channel 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23	RO	0x0	err_l_seq_di3 Error in the sequence of lines for vc3 and dt3
22	RO	0x0	err_l_seq_di2 Error in the sequence of lines for vc2 and dt2
21	RO	0x0	err_l_seq_di1 Error in the sequence of lines for vc1 and dt1
20	RO	0x0	err_l_seq_di0 Error in the sequence of lines for vc0 and dt0
19	RO	0x0	err_l_bndry_match_di3 Error matching line start with line end for vc3 and dt3
18	RO	0x0	err_l_bndry_match_di2 Error matching line start with line end for vc2 and dt2
17	RO	0x0	err_l_bndry_match_di1 Error matching line start with line end for vc1 and dt1
16	RO	0x0	err_l_bndry_match_di0 Error matching line start with line end for vc0 and dt0
15	RO	0x0	err_frame_data_vc3 Last received frame, in virtual channel 3, had at least one CRC error
14	RO	0x0	err_frame_data_vc2 Last received frame, in virtual channel 2 had at least one CRC error
13	RO	0x0	err_frame_data_vc1 Last received frame, in virtual channel 1, had at least one CRC error
12	RO	0x0	err_frame_data_vc0 Last received frame, in virtual channel 0, had at least one CRC error
11	RO	0x0	err_f_seq_vc3 Error in the sequence of lines for vc3 and dt3
10	RO	0x0	err_f_seq_vc2 Error in the sequence of lines for vc2 and dt2
9	RO	0x0	err_f_seq_vc1 Error in the sequence of lines for vc1 and dt1
8	RO	0x0	err_f_seq_vc0 Error in the sequence of lines for vc0 and dt0
7	RO	0x0	err_f_bndry_match_vc3 Error matching frame start with frame end for virtual channel 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RO	0x0	err_f_bndry_match_vc2 Error matching frame start with frame end for virtual channel 2
5	RO	0x0	err_f_bndry_match_vc1 Error matching frame start with frame end for virtual channel 1
4	RO	0x0	err_f_bndry_match_vc0 Error matching frame start with frame end for virtual channel 0
3	RO	0x0	phy_errsotsynchs_3 Start of transmission error on data lane 3(no synchronization achieved)
2	RO	0x0	phy_errsotsynchs_2 Start of transmission error on data lane 2 (no synchronization achieved)
1	RO	0x0	phy_errsotsynchs_1 Start of transmission error on data lane 1 (no synchronization achieved)
0	RO	0x0	phy_errsotsynchs_0 Start of transmission error on data lane 0 (no synchronization achieved)

**CSIHOST\_ERR2**

Address: Operational Base + offset (0x0024)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23	RO	0x0	err_l_seq_di7 Error in the sequence of lines for vc7 and dt7
22	RO	0x0	err_l_seq_di6 Error in the sequence of lines for vc6 and dt6
21	RO	0x0	err_l_seq_di5 Error in the sequence of lines for vc5 and dt5
20	RO	0x0	err_l_seq_di4 Error in the sequence of lines for vc4 and dt4
19	RO	0x0	err_l_bndry_match_di7 Error matching line start with line end for vc7 and dt7
18	RO	0x0	err_l_bndry_match_di6 Error matching line start with line end for vc6 and dt6
17	RO	0x0	err_l_bndry_match_di5 Error matching line start with line end for vc5 and dt5
16	RO	0x0	err_l_bndry_match_di4 Error matching line start with line end for vc4 and dt4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RO	0x0	err_id_vc3 Unrecognized or unimplemented data type detected in virtual channel 3
14	RO	0x0	err_id_vc2 Unrecognized or unimplemented data type detected in virtual channel 2
13	RO	0x0	err_id_vc1 Unrecognized or unimplemented data type detected in virtual channel 1
12	RO	0x0	err_id_vc0 Unrecognized or unimplemented data type detected in virtual channel 0
11	RO	0x0	vc3_err_ecc_corrected Header error detected and corrected on virtual channel 3
10	RO	0x0	vc2_err_ecc_corrected Header error detected and corrected on virtual channel 2
9	RO	0x0	vc1_err_ecc_corrected Header error detected and corrected on virtual channel 1
8	RO	0x0	vc0_err_ecc_corrected Header error detected and corrected on virtual channel 0
7	RO	0x0	phy_errsoths_3 Start of transmission error on data lane 3 (synchronization can still be achieved)
6	RO	0x0	phy_errsoths_2 Start of transmission error on data lane 2 (synchronization can still be achieved)
5	RO	0x0	phy_errsoths_1 Start of transmission error on data lane 1 (synchronization can still be achieved)
4	RO	0x0	phy_errsoths_0 Start of transmission error on data lane 0 (synchronization can still be achieved)
3	RO	0x0	phy_erresc_3 Escape entry error (ULPM) on data lane 3
2	RO	0x0	phy_erresc_2 Escape entry error (ULPM) on data lane 2
1	RO	0x0	phy_erresc_1 Escape entry error (ULPM) on data lane 1
0	RO	0x0	phy_erresc_0 Escape entry error (ULPM) on data lane 0

**CSIHOST\_MSK1**

Address: Operational Base + offset (0x0028)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28	RW	0x0	mask_err_ecc_double Mask for err_ecc_double
27	RW	0x0	mask_vc3_err_crc Mask for vc3_err_crc
26	RW	0x0	mask_vc2_err_crc Mask for vc2_err_crc
25	RW	0x0	mask_vc1_err_crc Mask for vc1_err_crc
24	RW	0x0	mask_vc0_err_crc Mask for vc0_err_crc
23	RW	0x0	mask_err_l_seq_di3 Mask for err_l_seq_di3
22	RW	0x0	mask_err_l_seq_di2 Mask for err_l_seq_di2
21	RW	0x0	mask_err_l_seq_di1 Mask for err_l_seq_di1
20	RW	0x0	mask_err_l_seq_di0 Mask for err_l_seq_di0
19	RW	0x0	mask_err_l_bndry_match_di3 Mask for err_l_bndry_match_di3
18	RW	0x0	mask_err_l_bndry_match_di2 Mask for err_l_bndry_match_di2
17	RW	0x0	mask_err_l_bndry_match_di1 Mask for err_l_bndry_match_di1
16	RW	0x0	mask_err_l_bndry_match_di0 Mask for err_l_bndry_match_di0
15	RW	0x0	mask_err_frame_data_vc3 Mask for err_frame_data_vc3
14	RW	0x0	mask_err_frame_data_vc2 Mask for err_frame_data_vc2
13	RW	0x0	mask_err_frame_data_vc1 Mask for err_frame_data_vc1
12	RW	0x0	mask_err_frame_data_vc0 Mask for err_frame_data_vc0
11	RW	0x0	mask_err_f_seq_vc3 Mask for err_f_seq_vc3
10	RW	0x0	mask_err_f_seq_vc2 Mask for err_f_seq_vc2
9	RW	0x0	mask_err_f_seq_vc1 Mask for err_f_seq_vc1
8	RW	0x0	mask_err_f_seq_vc0 Mask for err_f_seq_vc0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	mask_err_f_bndry_match_vc3 Mask for err_f_bndry_match_vc1
6	RW	0x0	mask_err_f_bndry_match_vc2 Mask for err_f_bndry_match_vc1
5	RW	0x0	mask_err_f_bndry_match_vc1 Mask for err_f_bndry_match_vc1
4	RW	0x0	mask_err_f_bndry_match_vc0 Mask for err_f_bndry_match_vc0
3	RW	0x0	mask_phy_errsotsynchs_3 Mask for phy_errsotsynchs_3
2	RW	0x0	mask_phy_errsotsynchs_2 Mask for phy_errsotsynchs_2
1	RW	0x0	mask_phy_errsotsynchs_1 Mask for phy_errsotsynchs_1
0	RW	0x0	mask_phy_errsotsynchs_0 Mask for phy_errsotsynchs_0

**CSIHOST\_MSK2**

Address: Operational Base + offset (0x002c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23	RO	0x0	err_l_seq_di7 Error in the sequence of lines for vc7 and dt7
22	RO	0x0	err_l_seq_di6 Error in the sequence of lines for vc6 and dt6
21	RO	0x0	err_l_seq_di5 Error in the sequence of lines for vc5 and dt5
20	RO	0x0	err_l_seq_di4 Error in the sequence of lines for vc4 and dt4
19	RO	0x0	err_l_bndry_match_di7 Error matching line start with line end for vc7 and dt7
18	RO	0x0	err_l_bndry_match_di6 Error matching line start with line end for vc6 and dt6
17	RO	0x0	err_l_bndry_match_di5 Error matching line start with line end for vc5 and dt5
16	RO	0x0	err_l_bndry_match_di4 Error matching line start with line end for vc4 and dt4
15	RO	0x0	err_id_vc3 Unrecognized or unimplemented data type detected in virtual channel 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14	RO	0x0	err_id_vc2 Unrecognized or unimplemented data type detected in virtual channel 2
13	RO	0x0	err_id_vc1 Unrecognized or unimplemented data type detected in virtual channel 1
12	RO	0x0	err_id_vc0 Unrecognized or unimplemented data type detected in virtual channel 0
11	RO	0x0	vc3_err_ecc_corrected Header error detected and corrected on virtual channel 3
10	RO	0x0	vc2_err_ecc_corrected Header error detected and corrected on virtual channel 2
9	RO	0x0	vc1_err_ecc_corrected Header error detected and corrected on virtual channel 1
8	RO	0x0	vc0_err_ecc_corrected Header error detected and corrected on virtual channel 0
7	RO	0x0	phy_errsoths_3 Start of transmission error on data lane 3 (synchronization can still be achieved)
6	RO	0x0	phy_errsoths_2 Start of transmission error on data lane 2 (synchronization can still be achieved)
5	RO	0x0	phy_errsoths_1 Start of transmission error on data lane 1 (synchronization can still be achieved)
4	RO	0x0	phy_errsoths_0 Start of transmission error on data lane 0 (synchronization can still be achieved)
3	RW	0x0	mask_phy_erresc_3 Mask for phy_erresc_3
2	RW	0x0	mask_phy_erresc_2 Mask for phy_erresc_2
1	RW	0x0	mask_phy_erresc_1 Mask for phy_erresc_1
0	RW	0x0	mask_phy_erresc_0 Mask for phy_erresc_0

**CSIHOST\_PHY\_TEST\_CTRL0**

Address: Operational Base + offset (0x0030)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	phy_testclk D-PHY test interface strobe signal It is used to clock TESTDIN bus into the D-PHY. In conjunction with TESTEN signal controls controls the operation selection
0	RW	0x0	phy_testclr D-PHY test interface clear It is used when active performs vendor specific interface initialization (active high)

**CSIHOST\_PHY\_TEST\_CTRL1**

Address: Operational Base + offset (0x0034)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved
16	RW	0x0	phy_testen D-PHY test interface operation selector: 1: configures address write operation on the falling edge of TESTCLK 0: configures a data write operation on the rising edge of TESTCLK
15:8	RO	0x00	phy_testdout D-PHY output 8-bit data bus for read-back and internal probing functionalities
7:0	RW	0x00	phy_testdin D-PHY test interface input 8-bit data bus for internal register programming and test functionlities access

**36.5 Application Notes**

## Chapter 37 MIPI Controller

### 37.1 Overview

The Display Serial Interface (DSI) is part of a group of communication protocols defined by the MIPI Alliance. The MIPI Controller is a digital core that implements all protocol functions defined in the MIPI DSI Specification. The MIPI Controller provides an interface between the system and the MIPI D-PHY, allowing the communication with a DSI-compliant display. The MIPI Controller supports one to four lanes for data transmission with MIPI D-PHY.

The MIPI Controller supports the following features:

- Compliant with MIPI Alliance standards
- Support the DPI interface color coding mappings into 24-bit Interface
  - 16 bits per pixel, configurations 1,2, and 3
  - 18 bits per pixel, configurations 1 and 2
  - 24 bits per pixel
- Programmable polarity of all DPI interface signals
- Extended resolutions beyond the DPI standard maximum resolution of 800x480 pixels:
  - Up to 2047 vertical active lines
  - Up to 63 vertical back porch lines
  - Up to 63 vertical front porch lines
  - Maximum resolution is limited by available DSI Physical link bandwidth which depends on the number of lanes and maximum speed per lane
- All commands defined in MIPI Alliance Specification for Display Command Set (DCS)
- Interface with MIPI D-PHY following PHY Protocol Interface (PPI), as defined in MIPI Alliance Specification for D-PHY
- Up to four D-PHY Data Lanes
- Bidirectional communication and escape mode support through data lane 0
- Transmission of all generic commands
- ECC and Checksum capabilities
- End of Transmission Packet(EOTP)
- Ultra Low-Power mode
- Fault recovery schemes

### 37.2 Block Diagram

The following diagram shows the MIPI Controller architecture.

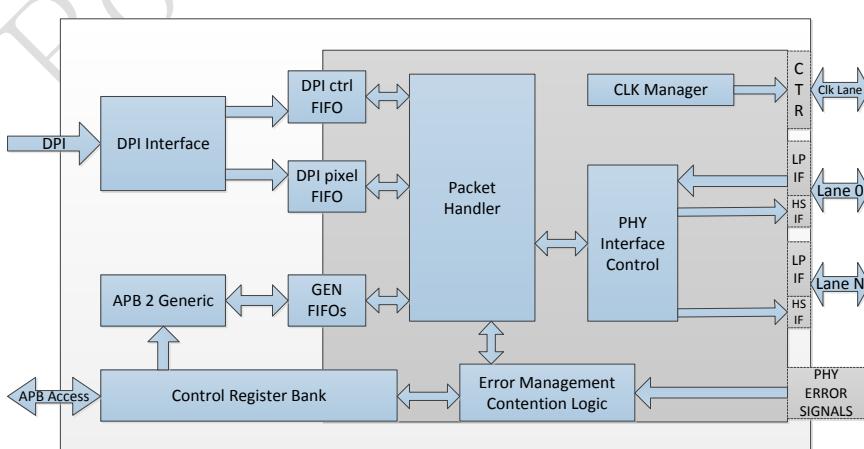


Fig. 37-1 MIPI Controller architecture

The DPI interface captures the data and control signals and conveys them to a FIFO for video

control signals and another one for pixel data. This data is then used to build Video packets, here in Video mode.

The Register Bank is accessible through a standard AMBA-APB slave interface, providing access to the MIPI Controller registers for configuration and control. There is also a fully programmable interrupt generator to inform the system about certain events.

The PHY Interface Control is responsible for managing the D-PHY PPI interface. It acknowledges the current operation and enables low-power transmission/reception or a high-speed transmission. It also performs data splitting between available D-PHY lanes for high-speed transmission.

The Packet Handler schedules the activities inside the link. It performs several functions based on the interfaces that are currently DPI and the video transmission mode that is used (burst mode or non-burst mode with sync pulse or sync events). It builds long or short packet generating correspondent ECC and CRC codes. This block also performs the following functions: Packet reception, Validation of packet header by checking the ECC, Header correction and notification for single-bit errors, Termination of reception, Multiple header error notification.

The APB-to-Generic block bridges the APB operations into FIFOs holding the Generic commands. The block interfaces with the following FIFOs: Command FIFO, Write payload FIFO, Read payload FIFO.

The Error Management notifies and monitors the error conditions on the DSI link. It controls the timers used to determine if a timeout condition occurred, performing an internal soft reset and triggering an interruption notification.

## **37.3 Function Description**

### **37.3.1 DPI interface function**

The DPI interface follows the MIPI DPI specification with pixel data bus width up to 24 bits. It is used to transmit the information in Video mode in which the transfers from the host processor to the peripheral take the form of a real-time pixel stream. This interface allows sending ShutDown (SD) and ColorMode (CM) commands, which are triggered directly by writing to the register of CFG\_MISC\_CON[2:1]. To transfer additional commands(for example, to initialize the display), use another interface such as APB Slave Generic Interface to complement the DPI interface.

The DPI interface captures the data and control signals and conveys them to the FIFO interfaces that transmit them to the DSI link. Two different streams of data are presented at the interface; video control signals and pixel data. Depending on the interface color coding, the pixel data is disposed differently throughout the dpipixdata bus. The following table shows the Interface pixel color coding.

Table 37-1 Color table

Signal Line	16-bit			18-bit		24-bit
	Config1	Config2	Config3	Config1	Config2	
dipiidata23	Not used	R7				
dipiidata22	Not used	R6				
dipiidata21	Not used	Not used	R4	Not used	R5	R5
dipiidata20	Not used	R4	R3	Not used	R4	R4
dipiidata19	Not used	R3	R2	Not used	R3	R3
dipiidata18	Not used	R2	R1	Not used	R2	R2
dipiidata17	Not used	R1	R0	R5	R1	R1
dipiidata16	Not used	R0	Not used	R4	R0	R0
dipiidata15	R4	Not used	Not used	R3	Not used	G7
dipiidata14	R3	Not used	Not used	R2	Not used	G6
dipiidata13	R2	G5	G5	R1	G5	G5
dipiidata12	R1	G4	G4	R0	G4	G4
dipiidata11	R0	G3	G3	G5	G3	G3
dipiidata10	G5	G2	G2	G4	G2	G2
dipiidata9	G4	G1	G1	G3	G1	G1
dipiidata8	G3	G0	G0	G2	G0	G0
dipiidata7	G2	Not used	Not used	G1	Not used	B7
dipiidata6	G1	Not used	Not used	G0	Not used	B6
dipiidata5	G0	Not used	B5	B5	B5	B5
dipiidata4	B4	B4	B4	B4	B4	B4
dipiidata3	B3	B3	B3	B3	B3	B3
dipiidata2	B2	B2	B2	B2	B2	B2
dipiidata1	B1	B1	B1	B1	B1	B1
dipiidata0	B0	B0	Not used	B0	B0	B0

The DPI interface can be configured to increase flexibility and promote correct usage of this interface for several systems. These configuration options are as follows:

- Polarity control: All the control signals are programmable to change the polarity depending on system requirements.

After the MIPI Controller reset, DPI waits for the first VSYNC active transition to start signal sampling, including pixel data, and preventing image transmission in the middle of a frame.

If interface pixel color coding is 18 bits and the 18-bit loosely packed stream is disabled, the number of lines programmed in the pixels per lines configuration is a multiple of four. This means that in this mode, the two LSBs in the configuration are always inferred as zero. The specification states that in this mode, the pixel line size should be a multiple of four.

### 37.3.2 APB Slave Generic Interface

The APB Slave interface allows the transmission of generic information in Command mode, and follows the proprietary register interface. Commands sent through this interface are not constrained to comply with the DCS specification, and can include generic commands described in the DSI specification as manufacturer-specific.

The MIPI Controller supports the transmission or write and read command mode packets as described in the DSI specification. These packets are built using the APB register access. The GEN\_PLD\_DATA register has two distinct functions based on the operation. Writing to this register sends the data as payload when sending a Command mode packet. Reading this register returns the payload of a read back operation. The GEN\_HDR register contains the Command mode packet header type and header data. Writing to this register triggers the transmission of the packet implying that for a long Command mode packet, the packet's payload needs to be written in advance in the GEN\_PLD\_DATA register.

The valid packets available to be transmitted through the Generic interface are as follows:

Generic Write Short Packet 0 Parameters  
Generic Write Short Packet 1 Parameters  
Generic Write Short Packet 2 Parameter  
Generic Write Short Packet 0 Parameter  
Generic Write Short Packet 1 Parameters  
Generic Write Short Packet 2 Parameter  
Maximum Read Packet Configuration  
Generic Long Write Packet  
DCS Write Short Packet 0 Parameter  
DCS Write Short Packet 1 Parameter  
DCS Write Short Packet 0 Parameter  
DCS Write Long Packet

A set of bits in the CMD\_PKT\_STATUS register report the status of the FIFOs associated with APB interface support.

Generic interface packets are always transported using one of the DSI transmission modes; Video mode or Command mode. If neither of these mode are selected, the packets are not transmitted through the link and the released FIFOs eventually get overflowed.

The transfer of packets through the APB bus is based on the following conditions:

The APB protocol defines that the write and read procedure takes two clock cycles each to be executed. This means that the maximum input data rate through the APB interfaces is always half the speed of the APB clock.

The data input bus has a maximum width of 32 bits. This allows for a relation to be defined between the input APB clock frequency and maximum bi rate achievable by the APB interface.

The DSI link bit rate when using solely APB is equal to (APB clock frequency) \*16 Mbps.

The bandwidth is dependent on the APB clock frequency; the available bandwidth increases with the clock frequency.

To drive the APB interface to achieve high bandwidth Command mode traffic transported by the DSI link, the MIPI Controller should operate in the Command mode only and the APB interface should be the only data source that is currently in use. Thus, the APB interface has the entire bandwidth of the DSI link and does not share it with any another input interface source.

The memory write commands require maximum throughout from the APB interface, because they contain the most amount of data conveyed by the DSI link. While writing the packet information, first write the payload of a given packet into the payload FIFO using the GEN\_PLD\_DATA register. When the payload data is for the command parameters, place the first byte to be transmitted in the least significant byte position of the APB data bus.

After writing the payload, write the packet header into the command FIFO. For more information and it should follow the pixel to byte conversion organization referred in the Annexure A of the DCS specification. The follow figures show how the pixel data should be organized in the APB data write bus. The memory write commands are conveyed in DCS long packets. DCS long packets are encapsulated in a DSI packet. The DSI included in the diagrams. In the follow figures, the Write Memory Command can be replaced by the DCS command Write Memory Start and Write Memory Continue.

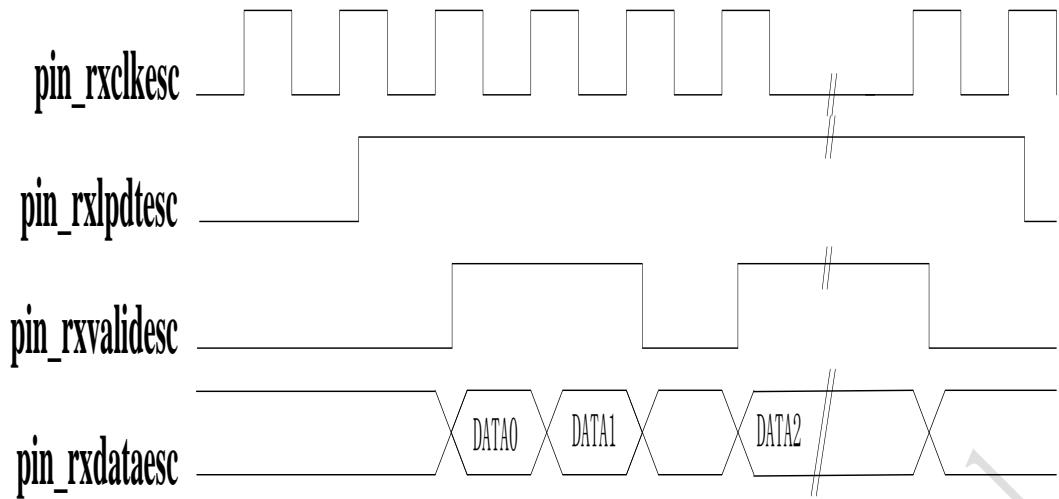


Fig. 37-2 24bpp APB Pixel to Byte Organization

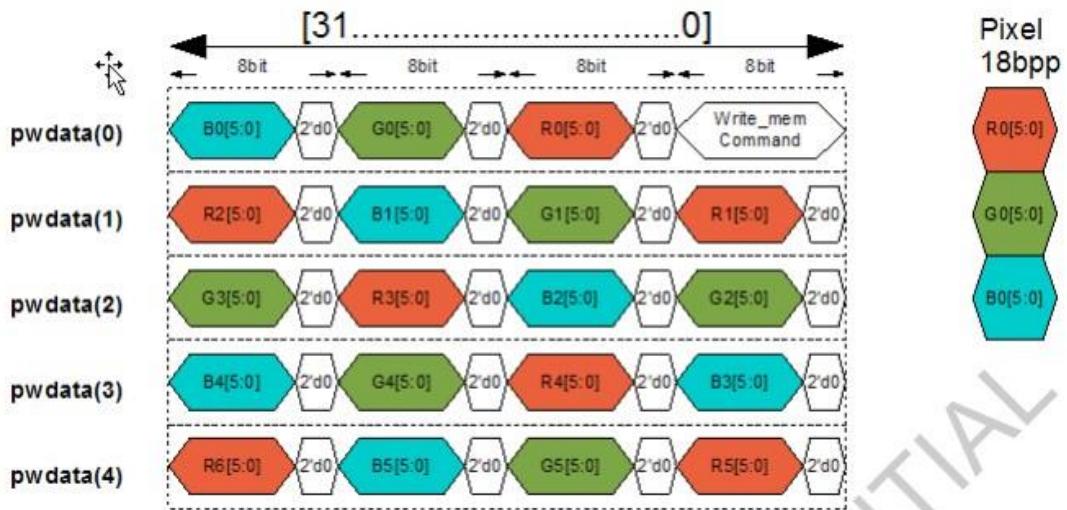


Fig. 37-3 18 bpp APB Pixel to Byte Organization

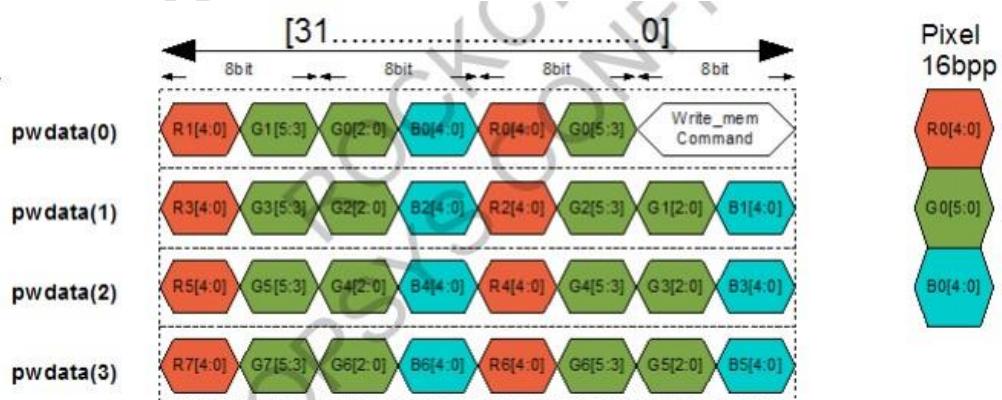


Fig. 37-4 16 bpp APB Pixel to Byte Organization

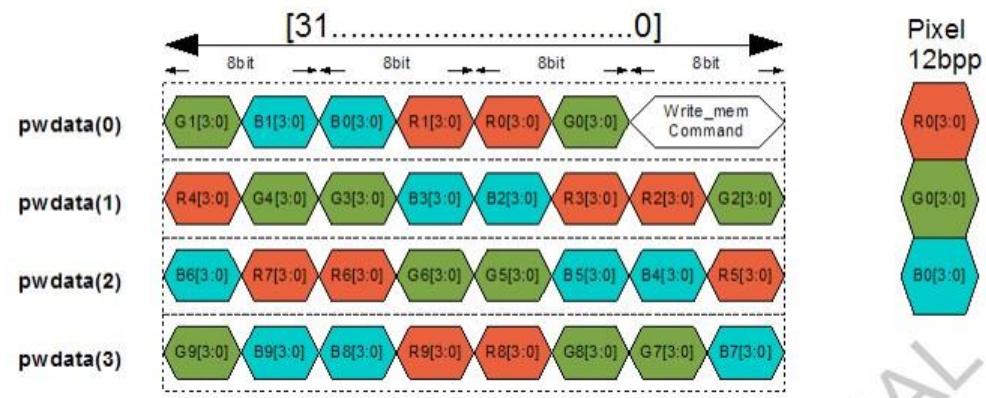


Fig. 37-5 12 bpp APB Pixel to Byte Organization

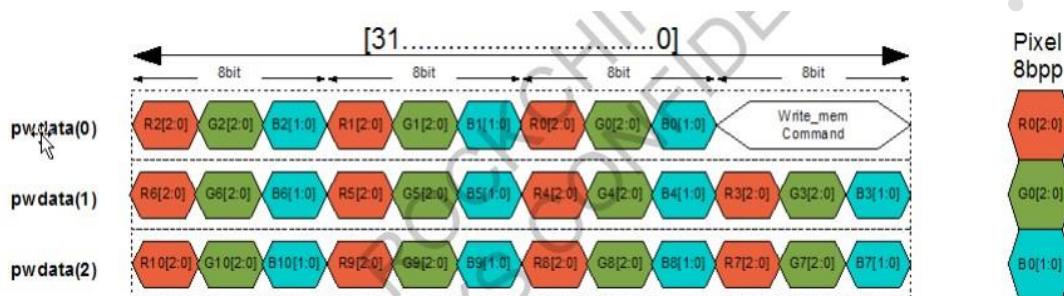


Fig. 37-6 8bpp APB Pixel to Byte Organization

### 37.3.3 Transmission of Commands in Video Mode

The MIPI Controller supports the transmission of commands, both in high-speed and low-power, while in Video mode. The DSI controller uses Blanking or Low-Power(BLLP) periods to transmit commands inserted through the APB Generic interface. Those periods correspond to the shaded areas of the following figure.

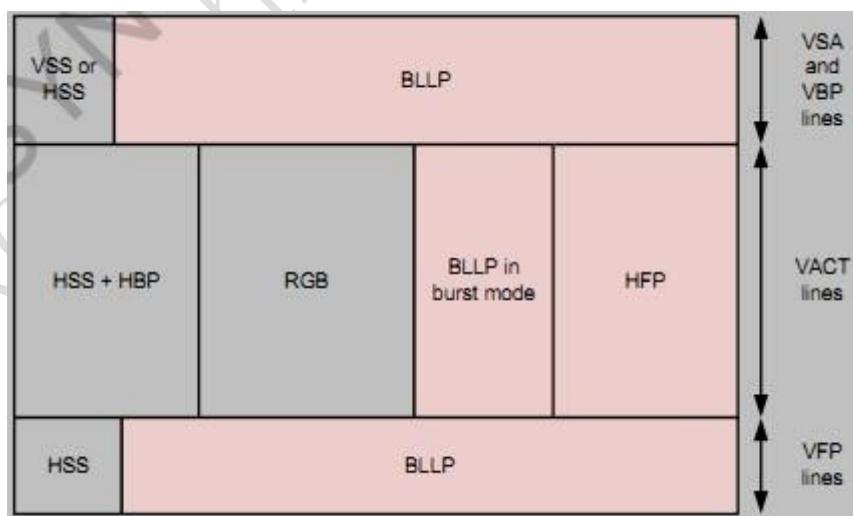


Fig. 37-7 Command Transmission Periods within the Image Area

Commands are transmitted in the blanking periods after the following packet/states:

- Vertical Sync Start (VSS) packets, if the Video Sync pulses are not enabled
- Horizontal Sync End (HSE) packets, in the VSA, VBP, and VFP regions
- Horizontal Sync Start (HSS) packets, if the Video Sync pulses are not enabled in the VSA,

- VBP, and VFP regions
- Horizontal Active (HACT) state

Only one command is transmitted per line, even in the case of the last line of a frame but one command is possible for each line.

The MIPI Controller avoids sending commands in the last line because it is possible that the last line is shorter than the other ones. For instance, the line time ( $tL$ ) could be half a cycle longer than the  $tL$  on the DPI interface, that is, each line in the frame taking half a cycle from time for the last line. This results in the last line being  $(1/2 \text{ cycle}) * (\text{number of lines} - 1)$  shorter than  $tL$ .

The dpicolorm and dpishutdn input signals are also able to trigger the sending of command packets. The commands are DSI data types Color Mode On, Color Mode Off, Shut Down Peripheral, and Turn on Peripheral. These commands are not sent in the VACT region. If the lpcmden bit of the VID\_MODE\_CFG register is 1, these commands are sent in LP mode. In LP mode, the ouvact\_lpcmd\_time field of the LP\_CMD\_TIM register is used to determine if these commands can be transmitted. It is assumed that outvact\_lpcmd\_time is greater than or equal to 4 bytes (number of bytes in a short packet), because the DWC\_mipi\_dsi\_host does not transmit these commands on the last line.

If the frame\_BTA\_ack field is set in the VID\_MODE\_CFG register, a BTA is generated by DWC\_mipi\_dsi\_host after the last line of a frame. This may coincide with a write command or a read command. In either case, the edpihalt signal is held asserted until an acknowledge has been received (control of the DSI bus is returned to the host).

If the lpcmden bit of the VID\_MODE\_CFG register is set to 1, the commands are sent in low-power in Video mode. In this case, it is necessary to calculate the time available, in bytes, to transmit a command in LP mode for Horizontal Front Porch (HFP), Vertical Sync Active (VSA), Vertical Back Porch (VBP), and Vertical Front Porch(VFP) regions.

The outvact\_lpcmd\_time field of the LP\_CMD\_TIM register indicates the time available (in bytes) to transmit a command in LP mode, based on the escape clock, on a line during the VSA, VBP, and the VFP

$$\text{Outvact\_lpcmd\_time} = (tL - (\text{Time to transmit HSS and HSE frames} + tHSA + \text{Time to enter and leave LP mode} + \text{Time to send the D-PHY LPDT command})) / \text{escape clock period} / 8 / 2$$

Where,

$tL$ =Line time

$tHSA$ =Time to send a short packet (for sync events) or time of the HAS pulse (for sync pulses)

In the above equation, division by eight is done to convert the time available to bytes and division by two is done because one bit is transmitted once in every two escape clock cycles.

The outvact\_lpcmd\_time filed can be compared directly with the size of the command to be transmitted to determine if there is enough time to transmit the command. The maximum size of a command that can be transmitted in LP mode is limited to 255 bytes by this field. This register must be programmed to a value greater than or equal to 4 bytes for the transmission of the DCTRL commands such as shutdown and colorm in LP mode.

Consider an example with  $12.6 \mu\text{s}$  per line and assume an escape clock frequency of 15 MHz. In this case, 189 escape clock cycles are available to enter and exit LP mode and transmit command. The following are assumed:

Sync pulses are not being transmitted

Two lane byte clock ticks are required to transmit a short packet

```
phy_lp2hs_time=16
phy_lp2p_time=20
```

In this example, a 11-byte command can be transmitted as follows:

$$\text{outvact\_lpcmd\_time} = (12.6\mu\text{s} - (2*10 \text{ ns}) - (16*10 \text{ ns}) - (20*10 \text{ ns}) - (8*66 \text{ ns})) / 66 \text{ ns} / 8 / 2 = 11 \text{ bytes}$$

The `invact_lpcmd_time` field of the `LP_CMD_TIM` register indicates the time available (in bytes) to transmit a command in LP mode (based on the escape clock) in the Vertical Active (VACT) region. This time is calculated as follows:

$$\text{Invact\_lpcmd\_time} = ((\text{tHFP} - \text{Time to enter and leave low-power mode} + \text{Blanking period before the HFP when in Burst mode} - \text{Time to send the D-PHY LPDT command}) / \text{escape clock period}) / 8$$

Where,

$$\text{tHFP} = \text{line time} - \text{tHSA} - \text{tHBP} - \text{tHACT}$$

$$\text{tHACT} = \text{vid\_pkt\_size} * \text{bits\_per\_pixel} * \text{lane\_byte\_clock\_period} / \text{num\_lanes}$$

The `invact_lpcmd_time` field can be compared directly with the size of the command to be transmitted to determine if there is time to transmit the command.

Consider an example where the refresh rate is 60 Hz. The number of lines is 1320 (typical). The tL in this case is 12.6μs. With a lane byte clock of 100 MHz, 1260 clock ticks are available to transmit a single frame. If 800 ticks are used for pixel data then 460 ticks (4.6μs) are available for Horizontal Sync Start (HSS), HFP, and HBP. Assuming that 2.3μs is available for HFP and the escape clock is 15MHz, only 34 LP clock ticks are available to enter LP, transmit a command, and return from LP mode. Approximately 12 escape clock ticks are required to enter and leave LP mode. Therefore, only 1 byte could be transmitted in this period.

A short packet (for example, generic short write) requires a minimum of 4 bytes. Therefore, in this example, commands are not sent in the VACT region. If Burst mode is enabled, more time is available to transmit commands in the VACT region. The following are assumed:

The controller is not in Burst mode

$$\begin{aligned}\text{phy\_lp2hs\_time} &= 16 \\ \text{phy\_lp2hs\_time} &= 16\end{aligned}$$

In this example `invact_lpcmd_time` is calculated as follows:

$$\text{Invact\_lpcmd\_time} = (2.3\mu\text{s} - (16*10 \text{ ns}) - (20*10 \text{ ns}) - (8*66 \text{ ns})) / 66 \text{ ns} / 8 = 2 \text{ bytes}$$

The `outvact_lpcmd_time` and `invact_lpcmd_time` fields allow a simple comparison to determine if a command can be transmitted in any of the BLLP periods.

Fig. 38-8 illustrates the meaning of `invact_lpcmd_time` and `outvact_lpcmd_time`, matching them with the shaded areas and the VACT region.

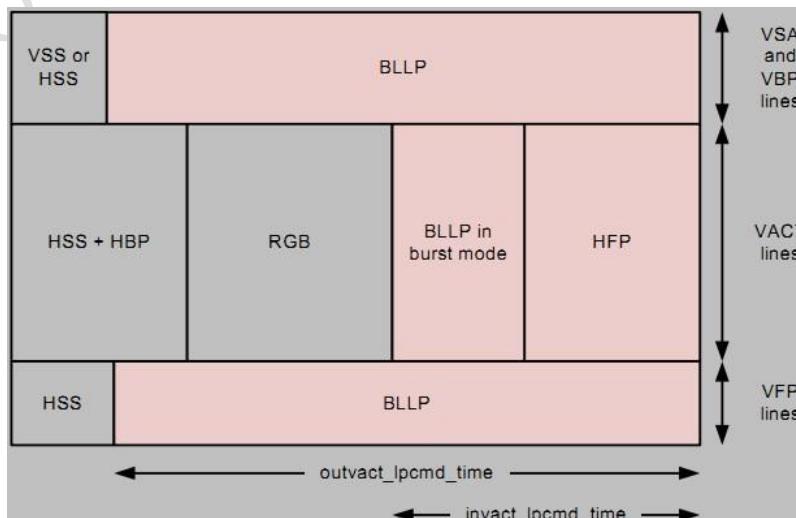


Fig. 37-8 Location in the Image Area

If the lpcmden bit of the VID\_MODE\_CFG register is 0, the commands are sent in high\_speed in Video Mode. In this case, the DWC\_mipi\_dsi\_host automatically determines the area where each command can be sent and no programming or calculation is required.

On read command Transmission, the max\_rd\_time field of the PHY\_TMR\_CFG register configures the maximum amount of time required to perform a read command in lane byte clock cycles.

The maximum time required to perform a read command in Lane byte clock cycles (max\_rd\_time) = Time to transmit the read command in LP mode + Time to enter and leave LP mode + Time to return the read data packet from the peripheral device.

The time to return the read data packet from the peripheral depends on the number of bytes read and the escape clock frequency of the peripheral; not the escape clock of the host. The max\_rd\_time field is used in both HS and LP mode to determine if there is time to complete a read command in a BLLP period.

In high-speed mode (lpcmden=0), max\_rd\_time is calculated as follows:

max\_rd\_time = phy\_hs2lp\_time + Time to return the read data packet from the peripheral device + phy\_hs2hs\_time

In low-power mode (lpcmden = 1), max\_rd\_time is calculated as follows:

max\_rd\_time = phy\_hs2lp\_time + LPDT command time + Read command time in LP mode + Time to return the data read from the peripheral device + phy\_lp2hs\_time

Where,

LPDT command time = (8\*Host escape clock period) / Lane byte clock period

Read command time in LP mode = (32 \* host escape clock period) / lane byte clock period

It is recommended to keep the maximum number of bytes read from the peripheral to a minimum to have sufficient time available to issue the read commands on a line. Ensure that max\_rd\_time\* Lane byte clock period is less than outvact\_lpcmd\_time \*8\*Escape clock period of the host.

Otherwise, the read commands are serviced on the last line of a frame and the edpihalt signal may be asserted. If it is necessary to read a large number of parameters (>16), increase the max\_rd\_time while the read command is being executed. When the read has completed, decrease the max\_rd\_time to a lower value.

## 37.4 Register Description

This section describes the control/status registers of the design.

### 37.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
MIPIC_VERSION	0x00000	W	0x3133302a	Version of the mipi controller
MIPIC_PWR_UP	0x00004	W	0x00000000	Core power-up
MIPIC_CLKMGR_CFG	0x00008	W	0x00000000	Configuration of the internal clock dividers
MIPIC_DPI_VCID	0x0000c	W	0x00000000	The DPI interface configuration.

Name	Offset	Size	Reset Value	Description
MIPIC_DPI_COLOR_CODING	0x00010	W	0x00000000	
MIPIC_DPI_CFG_POL	0x00014	W	0x00000000	
MIPIC_LP_CMD_TIM	0x00018	W	0x00000000	Low-power Command Timing Configuration Register.
MIPIC_PCKHDL_CFG	0x0002c	W	0x00000000	Packet handler configuration
MIPIC_GEN_VCID	0x00030	W	0x00000000	
MIPIC_MODE_CFG	0x00034	W	0x00000000	
MIPIC_VID_MODE_CFG	0x00038	W	0x00000000	Video mode configuration.
MIPIC_VID_PKT_SIZE	0x0003c	W	0x00000000	
MIPIC_VID_NUM_CHUNKS	0x00040	W	0x00000000	
MIPIC_VID_NULL_SIZE	0x00044	W	0x00000000	
MIPIC_VID_HSA_TIM	0x00048	W	0x00000000	Line timing configuration.
MIPIC_VID_HBP_TIM	0x0004c	W	0x00000000	
MIPIC_VID_HLINE_TIM	0x00050	W	0x00000000	
MIPIC_VID_VSA_LINES	0x00054	W	0x00000000	Vertical timing configuration.
MIPIC_VID_VBP_LINES	0x00058	W	0x00000000	
MIPIC_VID_VFP_LINES	0x0005c	W	0x00000000	
MIPIC_VID_VACTIVE_LINES	0x00060	W	0x00000000	
MIPIC_EDPI_CMD_SIZE	0x00064	W	0x00000000	
MIPIC_CMD_MODE_CFG	0x00068	W	0x00000000	Command mode configuration
MIPIC_GEN_HDR	0x0006c	W	0x00000000	Generic packet header configuration.
MIPIC_GEN_PLD_DATA	0x00070	W	0x00000000	Generic payload data in and out.
MIPIC_CMD_PKT_STATUS	0x00074	W	0x00000000	Command packet status
MIPIC_TO_CNT_CFG	0x00078	W	0x00000000	Timeout timers configuration

Name	Offset	Size	Reset Value	Description
MIPIC_HS_RD_TO_CNT	0x0007c	W	0x00000000	
MIPIC_LP_RD_TO_CNT	0x00080	W	0x00000000	
MIPIC_HS_WR_TO_CNT	0x00084	W	0x00000000	
MIPIC_LP_WR_TO_CNT	0x00088	W	0x00000000	
MIPIC_BTA_TO_CNT	0x0008c	W	0x00000000	
MIPIC_LPCLK_CTRL	0x00094	W	0x00000000	
MIPIC_PHY_TMR_LP_CLK_CFG	0x00098	W	0x00000000	
MIPIC_PHY_TMR_CFG	0x0009c	W	0x00000000	D-PHY timing configuration
MIPIC_PHY_RSTZ	0x000a0	W	0x00000000	D-PHY reset control
MIPIC_PHY_IF_CFG	0x000a4	W	0x00000000	D-PHY interface configuration
MIPIC_PHY_ULPS_CTRL	0x000a8	W	0x00000000	D-PHY PPI interface control
MIPIC_PHY_TX_TRIGERS	0x000ac	W	0x00000000	
MIPIC_PHY_STATUS	0x000b0	W	0x00000000	D-PHY PPI status interface
MIPIC_PHY_TST_CTRL0	0x000b4	W	0x00000001	
MIPIC_PHY_TST_CTRL1	0x000b8	W	0x00000000	
MIPIC_ERROR_ST0	0x000bc	W	0x00000000	Interrupt status register 0
MIPIC_ERROR_ST1	0x000c0	W	0x00000000	Interrupt status register 1
MIPIC_MSK0	0x000c4	W	0x00000000	Masks the interrupt generation triggerd by the ERROR_ST0 reg
MIPIC_MSK1	0x000c8	W	0x00000000	Masks the interrupt generation triggerd by the ERROR_ST1 reg
MIPIC_INT_FORCE0	0x000d8	W	0x00000000	
MIPIC_INT_FORCE1	0x000dc	W	0x00000000	
MIPIC_VID_SHADOW_CTRL	0x00100	W	0x00000000	
MIPIC_Copy0_DPI_VCID	0x0010c	W	0x00000000	The DPI interface configuration.

Name	Offset	Size	Reset Value	Description
MIPIC_Copy0 DPI_COLOR_CODING	0x00110	W	0x00000000	
MIPIC_Copy0 LP_CMD_TIM	0x00118	W	0x00000000	Low-power Command Timing Configuration Register.
MIPIC_Copy0 VID_MODE_CFG	0x00138	W	0x00000000	Video mode configuration.
MIPIC_Copy0 VID_PKT_SIZE	0x0013c	W	0x00000000	
MIPIC_Copy0 VID_NUM_CHUNKS	0x00140	W	0x00000000	
MIPIC_Copy0 VID_NULL_SIZE	0x00144	W	0x00000000	
MIPIC_Copy0 VID_HSA_TIME	0x00148	W	0x00000000	Line timing configuration.
MIPIC_Copy0 VID_HBP_TIME	0x0014c	W	0x00000000	
MIPIC_Copy0 VID_HLINE_TIME	0x00150	W	0x00000000	
MIPIC_Copy0 VID_VSA_LINES	0x00154	W	0x00000000	Vertical timing configuration.
MIPIC_Copy0 VID_VBP_LINES	0x00158	W	0x00000000	
MIPIC_Copy0 VID_VFP_LINES	0x0015c	W	0x00000000	
MIPIC_Copy0 VID_VACTIVE_LINES	0x00160	W	0x00000000	

Notes: Size: **B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**-WORD (32 bits) access

### 37.4.2 Detail Register Description

#### MIPIC\_VERSION

Address: Operational Base + offset (0x00000)

Version of the mihi controller

Bit	Attr	Reset Value	Description
31:0	RO	0x3133302a	version indicates the version of the mihi_controller

#### MIPIC\_PWR\_UP

Address: Operational Base + offset (0x00004)

Core power-up

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	shutdownz This bit indicates the core power-up or the reset 0-Reset 1-Power-up

**MIPIC\_CLKMGR\_CFG**

Address: Operational Base + offset (0x00008)

Configuration of the internal clock dividers

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:8	RW	0x00	TO_CLK_DIVISION This field indicates the division factor for the Time Out clock used as the timing unit in the configuration of HS to LP and LP to HS transition error.
7:0	RW	0x00	TX_ESC_CLK_DIVISION Field0000 Abstract This field indicates the division factor for the TX_Escape clock source(lanebyteclk).The value 0 and 1 stop the TX_ESC clock generation

**MIPIC\_DPI\_VCID**

Address: Operational Base + offset (0x0000c)

The DPI interface configuration.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1:0	RW	0x0	dpi_vid This field configures the DPI virtual channel id that is indexed to the Video mode packets.

**MIPIC\_DPI\_COLOR\_CODING**

Address: Operational Base + offset (0x00010)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8	RW	0x0	en18_loosely When set to 1, this bit enables 18 loosely packed pixel stream.
7:4	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	dpi_color_coding This field configures the DPI color coding as follows: 000:16bit configuration 1 001:16bit configuration 2 010:16bit configuration 3 011:18bit configuration 1 100:18bit configuration 2 101:24bit

**MIPIC\_DPI\_CFG\_POL**

Address: Operational Base + offset (0x00014)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4	RW	0x0	colorm_active_low When set to 1, this bit configures the color mode pin as active low
3	RW	0x0	shutd_active_low When set to 1, this bit configures the shut down pin as active low
2	RW	0x0	hsync_active_low When set to 1, this bit configures the horizontal synchronism pin as active low.
1	RW	0x0	vsync_active_low When set to 1, this bit configures the vertical synchronism pin as active low
0	RW	0x0	dataen_active_low When set to 1, this bit configures the data enable pin as active low

**MIPIC\_LP\_CMD\_TIM**

Address: Operational Base + offset (0x00018)

Low-power Command Timing Configuration Register.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23:16	RW	0x00	outvact_lpcmd_time outside VACT region command time.This field configures the time available to transmit a command in low-power mode.The time value is expressed in a number of bytes format.The number of bytes represents the maximum size of a packet that can fit in a line during the VSA,VBP, and VFP region. This field must be configured with a value greater than or equal to four bytes to allow the transmission of the DCTRL commands such as shutdown and colorm in low-power mode.
15:8	RO	0x0	reserved
7:0	RW	0x00	invact_lpcmd_time Inside VACT region command time.This field configures the time available to transmit a command in low-power mode.The time value is expressed in a number of bytes format.The number of bytes represents the maximum size of the packet that can fit a line during the VACT region.

**MIPIC\_PCKHDL\_CFG**

Address: Operational Base + offset (0x0002c)

Packet handler configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4	RW	0x0	en_CRC_rx When set to 1, this bit enables the CRC reception and error reporting
3	RW	0x0	en_ECC_rx When set to 1, this bit enables the ECC reception, error correction, and reporting
2	RW	0x0	en_BTA When set to 1, this bit enables the Bus Turn-Around(BTA) request.
1	RW	0x0	en_EOTP_rx Field0000 Abstract When set to 1, this bit enables the EOTP reception
0	RW	0x0	en_EOTP_tx Field0000 Abstract When set to 1, this bit enables the EOTP transmission

**MIPIC\_GEN\_VCID**

Address: Operational Base + offset (0x00030)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1:0	RW	0x0	gen_vcid_rx the Generic interface read-back virtual channel identification

**MIPIC\_MODE\_CFG**

Address: Operational Base + offset (0x00034)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	en_video_mode When set to 1, this bit enables the DPI Video mode transmission.

**MIPIC\_VID\_MODE\_CFG**

Address: Operational Base + offset (0x00038)

Video mode configuration.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	vpg_orientation This field indicates the color bar orientation as follows: 0:Vertical mode 1:Horizontal mode
23:21	RO	0x0	reserved
20	RW	0x0	vpg_mode This field is to select the pattern 0:Color bar(horizontal or vertical) 1:BER pattern(vertical only)
19:17	RO	0x0	reserved
16	RW	0x0	vpg_en When set to 1, this bit enables the video mode pattern generator
15	RW	0x0	lpcmden When set to 1, this bit enables the command transmission only in low-power mode
14	RW	0x0	frame_BTA_ack When set to 1, this bit enables the request for an acknowledge response at the end of a frame

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13	RW	0x0	en_lp_hfp When set to 1, this bit enables the return to low-power inside the HFP period when timing allows.
12	RW	0x0	en_lp_hbp When set to 1, this bit enables the return to low-power inside the HBP period when timing allows.
11	RW	0x0	en_lp_vact When set to 1, this bit enables the return to low-power inside the VACT period when timing allows.
10	RW	0x0	en_lp_vfp When set to 1, this bit enables the return to low-power inside the VFP period when timing allows.
9	RW	0x0	en_lp_vbp When set to 1, this bit enables the return to low-power inside the VBP period when timing allows.
8	RW	0x0	en_lp_vsa When set to 1, this bit enables the return to low-power inside the VSA period when timing allows.
7:2	RO	0x0	reserved
1:0	RW	0x0	vid_mode_type This field indicates the video mode transmission type as follows: 00: Non-burst with sync pulses 01: Non-burst with sync events 10 and 11: Burst with sync pulses

**MIPIC\_VID\_PKT\_SIZE**

Address: Operational Base + offset (0x0003c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13:0	RW	0x0000	vid_pkt_size This field configures the number of pixels on a single video packet. If you use the 18-bit mode and do not enable loosely packed stream, this value must be a multiple of 4.

**MIPIC\_VID\_NUM\_CHUNKS**

Address: Operational Base + offset (0x00040)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12:0	RW	0x0000	num_chunks This field configures the number of chunks to be transmitted during a line period(a chunk is a video packet or a null packet)

**MIPIC\_VID\_NULL\_SIZE**

Address: Operational Base + offset (0x00044)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12:0	RW	0x0000	null_pkt_size This field configures the number of bytes in a null packet

**MIPIC\_VID\_HSA\_TIME**

Address: Operational Base + offset (0x00048)

Line timing configuration.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	hsa_time This field configures the Horizontal Synchronism Active period in lane byte clock cycles.

**MIPIC\_VID\_HBP\_TIME**

Address: Operational Base + offset (0x0004c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	hbp_time This field configures the Horizontal Back Porch period in lane byte clock cycles

**MIPIC\_VID\_HLINE\_TIME**

Address: Operational Base + offset (0x00050)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	RO	0x0	reserved
14:0	RW	0x0000	hline_time This field configures the size of the total lines counted in lane byte cycles.

**MIPIC\_VID\_VSA\_LINES**

Address: Operational Base + offset (0x00054)

Vertical timing configuration.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9:0	RW	0x000	vsa_lines This field configures the Vertical Synchronism Active period measured in number of horizontal lines.

**MIPIC\_VID\_VBP\_LINES**

Address: Operational Base + offset (0x00058)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9:0	RW	0x000	vbp_lines This field configures the Vertical Back Porch period measured in horizontal lines.

**MIPIC\_VID\_VFP\_LINES**

Address: Operational Base + offset (0x0005c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9:0	RW	0x000	vfp_lines This field configures the Vertical Front Porch period measured in horizontal lines.

**MIPIC\_VID\_VACTIVE\_LINES**

Address: Operational Base + offset (0x00060)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13:0	RW	0x0000	v_active_line This field configures the Vertical Active period measured in horizontal lines.

**MIPIC\_EDPI\_CMD\_SIZE**

Address: Operational Base + offset (0x00064)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	edpi_allowed_cmd_size This field configures the maximum allowed size for an eDPI write memory command, measured in pixels. Automatic partitioning of data obtained from eDPI is permanently enabled.

**MIPIC\_CMD\_MODE\_CFG**

Address: Operational Base + offset (0x00068)

Command mode configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	max_rd_pkt_size This bit configures the maximum read packet size command transmission type: 0:High-speed 1:Low-power
23:20	RO	0x0	reserved
19	RW	0x0	dcs_lw_tx This bit configures the DCS long write packet command transmission type: 0:high-speed 1:low-power
18	RW	0x0	dcs_sr_0p_tx This bit configures the DCS short read packet with zero parameter command transmission type: 0:High-speed 1:Low-power
17	RW	0x0	dcs_sw_1p_tx This bit configures the DCS short write packet with one parameter command transmission type: 0:High-speed 1:Low-power
16	RW	0x0	dcs_sw_0p_tx This bit configures the DCS short write packet with zero parameter command transmission type: 0:High-speed 1:Low-power
15	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14	RW	0x0	gen_lw_tx This bit configures the Generic long write packet command transmission type: 0:high-speed 1:low-power
13	RW	0x0	gen_sr_2p_tx This bit configures the Generic short read packet with two parameter command transmission type: 0:High-speed 1:Low-power
12	RW	0x0	gen_sr_1p_tx This bit configures the Generic short read packet with one parameter command transmission type: 0:High-speed 1:Low-power
11	RW	0x0	gen_sr_0p_tx This bit configures the Generic short read packet with zero parameter command transmission type: 0:High-speed 1:Low-power
10	RW	0x0	gen_sw_2p_tx This bit configures the Generic short write packet with two parameter command transmission type: 0:High-speed 1:Low-power
9	RW	0x0	gen_sw_1p_tx This bit configures the Generic short write packet with one parameter command transmission type: 0:High-speed 1:Low-power
8	RW	0x0	gen_sw_0p_tx This bit configures the Generic short write packet with zero parameter command transmission type: 0:High-speed 1:Low-power
7:2	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	ack_rqst_en When set to 1, this bit enables the acknowledge request after each packet transmission
0	RW	0x0	tear_fx_en When set to 1, this bit enables the tearing effect acknowledge request

**MIPIC\_GEN\_HDR**

Address: Operational Base + offset (0x0006c)

Generic packet header configuration.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RO	0x0	reserved
23:16	RW	0x00	gen_WC_MSbyte This field configures the most significant byte of the header packet's Word count for long packets or data 1 for short packets.
15:8	RW	0x00	gen_WC_LSbyte This field configures the least significant byte of the header packet's Word count for long packets or data 0 for short packets.
7:6	RW	0x0	gen_VC This field configures the virtual channel id of the header packet.
5:0	RW	0x00	gen_DT This field configures the packet data type of the header packet

**MIPIC\_GEN\_PLD\_DATA**

Address: Operational Base + offset (0x00070)

Generic payload data in and out.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	gen_pld_b4 This field indicates byte 4 of the packet payload.
23:16	RW	0x00	gen_pld_b3 This field indicates byte 3 of the packet payload.
15:8	RW	0x00	gen_pld_b2 This field indicates byte 2 of the packet payload.
7:0	RW	0x00	gen_pld_b1 This field indicates byte 1 of the packet payload.

**MIPIC\_CMD\_PKT\_STATUS**

Address: Operational Base + offset (0x00074)

Command packet status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7	RW	0x0	reserved reserved
6	RW	0x0	gen_rd_cmd_busy This bit is set when a read command is issued and cleared when the entire response is stored in the FIFO
5	RW	0x0	gen_pld_r_full This bit indicates the full status of the generic read payload FIFO Vaule after reset:0x0
4	RO	0x0	gen_pld_r_empty This bit indicates the empty status of the generic read payload FIFO Vaule after reset:0x1
3	RO	0x0	gen_pld_w_full This bit indicates the full status of the generic write payload FIFO Vaule after reset:0x0
2	RO	0x0	gen_pld_w_empty This bit indicates the empty status of the generic write payload FIFO Vaule after reset:0x1
1	RO	0x0	gen_cmd_full This bit indicates the full status of the generic command FIFO Vaule after reset:0x0
0	RO	0x0	gen_cmd_empty This bit indicates the empty status of the generic command FIFO Vaule after reset:0x1

**MIPIC\_TO\_CNT\_CFG**

Address: Operational Base + offset (0x00078)

Timeout timers configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	hstx_to_cnt This field configures the timeout counter that triggers a high-speed transmission timeout contention detection(measured in TO_CLK_DIVISION cycles)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	lpx_to_cnt This field configures the timeout counter that triggers a low-power reception timeout contention detection(measured in TO_CLK_DIVISION cycles)

**MIPIC\_HS\_RD\_TO\_CNT**

Address: Operational Base + offset (0x0007c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	hs_rd_to_cnt This field sets a period for which the MIPI Controller keeps the link still,after sending a high-speed read operation.This period is measured in cycles of lanebyteclk.The counting starts when the D-PHY enters the Stop state and causes no interrupts.

**MIPIC\_LP\_RD\_TO\_CNT**

Address: Operational Base + offset (0x00080)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	lp_rd_to_cnt This field sets a period for which MIPI Controller keeps the link still,after sending a low-power read operation.This period is measured in cycles of lanebyteclk.The counting starts when the D-PHY enters the Stop state and causes no interrupts.

**MIPIC\_HS\_WR\_TO\_CNT**

Address: Operational Base + offset (0x00084)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
24	RW	0x0	presp_to_mode When set to 1, this bit ensures that the peripheral response timeout caused by hs_wr_to_cnt is used only once per eDPI frame, when both the following conditions are met: .dipi_sync_edpiwms has risen and fallen .packets originated from eDPI have been transmitted and its FIFO is empty again.
23:16	RO	0x0	reserved
15:0	RW	0x0000	hs_wr_to_cnt This field sets a period for which the MIPI Controller keeps the link inactive after sending a high-speed write operation. This period is measured in cycles of lanebyteclk. The counting starts when the D-PHY enters the Stop state and causes no interrupts.

**MIPIC\_LP\_WR\_TO\_CNT**

Address: Operational Base + offset (0x00088)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	lp_wr_to_cnt This field sets a period for which the DSI Controller keeps the link still, after sending a low-power write operation. This period is measured in cycles of lanebyteclk. The counting starts when the D-PHY enters the Stop state and causes no interrupts.

**MIPIC\_BTA\_TO\_CNT**

Address: Operational Base + offset (0x0008c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	bta_to_cnt This field sets a period for which the DSI Controller keeps the link still, after completing a Bus Turn-Around. This period is measured in cycles of lanebyteclk. The counting starts when the D-PHY enters the Stop state and causes no interrupts.

**MIPIC\_LPCLK\_CTRL**

Address: Operational Base + offset (0x00094)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	auto_clklane_ctrl This bit enables the automatic mechanism to stop providing clock in the clock lane when time allows.
0	RW	0x0	phy_txrequestclkhs This bit controls the D-PHY PPI tx requestclkhs signal

### **MIPIC\_PHY\_TMR\_LPCLK\_CFG**

Address: Operational Base + offset (0x00098)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RO	0x0	reserved
25:16	RW	0x000	phy_hs2lp_time This field configures the maximum time that the PHY takes to go from high-speed to low-power transmission measured in lane byte clock cycles.(clock lane)
15:10	RO	0x0	reserved
9:0	RW	0x000	phy_lp2hs_time This field configures the maximum time that the PHY takes to go from low-power to high-speed transmission measured in lane byte clock cycles.(clock lane)

### **MIPIC\_PHY\_TMR\_CFG**

Address: Operational Base + offset (0x0009c)

D-PHY timing configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	phy_hs2lp_time This field configures the maximum time that the PHY takes to go from high-speed to low-power transmission measured in lane byte clock cycles.
23:16	RW	0x00	phy_lp2hs_time This field configures the maximum time that the PHY takes to go from low-power to high-speed transmission measured in lane byte clock cycles.
15	RW	0x0	reserved reserved for future use

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
14:0	RW	0x0000	max_rd_time This field configures the maximum time required to perform a read command in lane byte clock cycles. This register can only be modified when read commands are not in progress.

**MIPIC\_PHY\_RSTZ**

Address: Operational Base + offset (0x000a0)

D-PHY reset control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3	RW	0x0	phy_forcepll When the D-PHY is in ULPS, this bit enables the D-PHY PLL
2	RW	0x0	phy_enableclk When set to 1, this bit enables the D-PHY Clock Lane Module
1	RW	0x0	phy_rstz When set to 0, this bit places the digital section of the D-PHY in the reset state
0	RW	0x0	phy_shutdownnz When set to 0, this bit places the D-PHY macro in power-down state

**MIPIC\_PHY\_IF\_CFG**

Address: Operational Base + offset (0x000a4)

D-PHY interface configuration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:8	RW	0x00	phy_stop_wait_time This field configures the minimum wait period to request a high-speed transmission after the Stop state is accounted in clock lane cycles.
7:2	RO	0x0	reserved
1:0	RW	0x0	n_lanes This field configures the number of active data lanes: 00: One data lane(lane 0) 01: Two data lane(lanes 0 and 1) 10: Three data lanes(lanes 0,1, and 2) 11: Four data lanes(lanes 0,1,2, and 3)

**MIPIC\_PHY\_ULPS\_CTRL**

Address: Operational Base + offset (0x000a8)

## D-PHY PPI interface control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3	RW	0x0	phy_txexitulpslan ULPS mode Exit on all active data lanes
2	RW	0x0	phy_txrequlpslan ULPS mode Request on all active data lanes
1	RW	0x0	phy_txexitulpsclk ULPS mode Exit on clock lane
0	RW	0x0	phy_txrequlpsclk ULPS mode Request on clock lane

**MIPIC\_PHY\_TX\_TRIGGER**

Address: Operational Base + offset (0x000ac)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3:0	RW	0x0	phy_tx_triggers This field controls the trigger transmissions.

**MIPIC\_PHY\_STATUS**

Address: Operational Base + offset (0x000b0)

## D-PHY PPI status interface

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12	RO	0x0	ulpsactivenot3lane This bit indicates the status of ulpsactivenot3lane D-PHY signal
11	RO	0x0	phystopstate3lane This bit indicates the status of phystopstate3lane D-PHY signal
10	RO	0x0	ulpsactivenot2lane This bit indicates the status of ulpsactivenot2lane D-PHY signal
9	RO	0x0	phystopstate2lane This bit indicates the status of phystopstate2lane D-PHY signal
8	RO	0x0	ulpsactivenot1lane This bit indicates the status of ulpsactivenot1lane D-PHY signal
7	RO	0x0	phystopstate1lane This bit indicates the status of phystopstate1lane D-PHY signal
6	RW	0x0	rxulpsesc0lane This bit indicates the status of rxulpsesc0lane D-PHY signal

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RO	0x0	ulpsactivenot0lane This bit indicates the status of ulpsactivenot0lane D-PHY signal
4	RO	0x0	phystopstate0lane This bit indicates the status of phystopstate0lane D-PHY signal
3	RO	0x0	phyulpsactivenotclk This bit indicates the status of phyulpsactivenotclk D-PHY signal
2	RO	0x0	phystopstateclklane This bit indicates the status of phystopstateclklane D-PHY signal
1	RO	0x0	phydirection This bit indicates the status of phydirection D-PHY signal
0	RO	0x0	phylock This bit indicates the status of phylock D-PHY signal

**MIPIC\_PHY\_TST\_CTRL0**

Address: Operational Base + offset (0x000b4)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	phy_testclk This bit is used to clock the TESTDIN bus into the D-PHY
0	RW	0x1	phy_testclr PHY test interface clear(active high)

**MIPIC\_PHY\_TST\_CTRL\_1**

Address: Operational Base + offset (0x000b8)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved
16	RW	0x0	phy_testen PHY test interface operation selector: 1:The address write operation is set on the falling edge of the testclk signal 0:The data write operation is set on the rising edge of the testclk signal
15:8	RW	0x00	phy_testdout PHY output 8-bit data bus for read-back and internal probing functionalities

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	RW	0x00	phy_testdin PHY test interface input 8-bit data bus for internal register programming and test functionalities access

**MIPIC\_ERROR\_ST0**

Address: Operational Base + offset (0x000bc)

Interrupt status register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:21	RO	0x0	reserved
20	RO	0x0	dphy_errors_4 This bit indicates LP1 contention error ErrContentionLP1 from Lane 0
19	RO	0x0	dphy_errors_3 This bit indicates LP0 contention error ErrContentionLP0 from Lane 0
18	RO	0x0	dphy_errors_2 This bit indicates control error ErrControl from Lane 0
17	RO	0x0	dphy_errors_1 This bit indicates ErrSyncEsc low-power data transmission synchronization error from Lane 0
16	RO	0x0	dphy_errors_0 This bit indicates ErrEsc escape entry error from Lane 0
15	RO	0x0	ack_with_err_15 This bit retrieves the DSI protocol violation from the Display Acknowledge error report
14	RO	0x0	ack_with_err_14 This bit retrieves the reserved(specific to device) from the Display Acknowledge error report
13	RO	0x0	ack_with_err_13 This bit retrieves the invalid transmission length from the Display Acknowledge error report
12	RO	0x0	ack_with_err_12 This bit retrieves the DSI VC ID Invalid from the Display Acknowledge error report
11	RO	0x0	ack_with_err_11 This bit retrieves the not recognized DSI data type from the Display Acknowledge error report

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
10	RO	0x0	ack_with_err_10 This bit retrieves the checksum error(long packet only) from the Display Acknowledge error report
9	RO	0x0	ack_with_err_9 This bit retrieves the ECC error,multi-bit(detected and corrected) from the Display Acknowledge error report
8	RO	0x0	ack_with_err_8 This bit retrieves the ECC error,single-bit(detected and corrected) from the Display Acknowledge error report
7	RO	0x0	ack_with_err_7 This bit retrieves the reserved(specific to device) error from the Display Acknowledge error report
6	RO	0x0	ack_with_err_6 This bit retrieves the False Control error from the Display Acknowledge error report
5	RO	0x0	ack_with_err_5 This bit retrieves the HS Receive Timeout error from the Display Acknowledge error report
4	RO	0x0	ack_with_err_4 This bit retrieves the LP Transmit Sync error from the Display Acknowledge error report
3	RO	0x0	ack_with_err_3 This bit retrieves the Escape Mode Entry command error from the Display Acknowledge error report
2	RO	0x0	ack_with_err_2 This bit retrieves the EoT sync error from the Display Acknowledge error report
1	RO	0x0	ack_with_err_1 This bit retrieves the SoT Sync error from the Display Acknowledge error report
0	RO	0x0	ack_with_err_0 This bit retrieves the SoT error from the Display Acknowledge error report

**MIPIC\_ERROR\_ST1**

Address: Operational Base + offset (0x000c0)

Interrupt status register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12	RO	0x0	gen_pld_recv_err This bit indicates that during a generic interface packet read back, the payload FIFO becomes full and the received data is corrupted.
11	RO	0x0	gen_pld_rd_err This bit indicates that during a DCS read data, the payload FIFO becomes empty and the data sent to the interface is corrupted
10	RO	0x0	gen_pld_send_err This bit indicates that during a Generic interface packet build, the payload FIFO becomes empty and corrupt data is sent.
9	RO	0x0	gen_pld_wr_err This bit indicates that the system tried to write a payload data through the Generic interface and the FIFO is full. Therefore, the command is not written.
8	RO	0x0	gen_cmd_wr_err This bit indicates that the system tried to write a command through the Generic interface and the FIFO is full. Therefore, the command is not written.
7	RO	0x0	dpi_pld_wr_err This bit indicates that during a DPI pixel line storage, the payload FIFO becomes full and the data stored is corrupted.
6	RO	0x0	eopt_err This bit indicates that the EOTP packet is not received at the end of the incoming peripheral transmission.
5	RO	0x0	pkt_size_err This bit indicates that the packet size error is detected during the packet reception.
4	RO	0x0	crc_err This bit indicates that the CRC error is detected in a received packet.
3	RO	0x0	ecc_multi_err This bit indicates that the ECC multiple error is detected and corrected in a received packet.
2	RO	0x0	ecc_single_err This bit indicates that the ECC single error is detected and corrected in a received packet.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RO	0x0	to_lp_rx This bit indicates that the low-power reception timeout counter reached the end and contention detection is detected.
0	RO	0x0	to_hs_tx This bit indicates that the high-speed transmission timeout counter reached the end and contention detection is detected.

**MIPIC\_MSK0**

Address: Operational Base + offset (0x000c4)

Masks the interrupt generation triggered by the ERROR\_ST0 reg

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:21	RO	0x0	reserved
20	RW	0x0	dphy_errors_4 This bit indicates LP1 contention error ErrContentionLP1 from Lane 0
19	RW	0x0	dphy_errors_3 This bit indicates LP0 contention error ErrContentionLP0 from Lane 0
18	RW	0x0	dphy_errors_2 This bit indicates control error ErrControl from Lane 0
17	RW	0x0	dphy_errors_1 This bit indicates ErrSyncEsc low-power data transmission synchronization error from Lane 0
16	RW	0x0	dphy_errors_0 This bit indicates ErrEsc escape entry error from Lane 0
15	RW	0x0	ack_with_err_15 This bit retrieves the DSI protocol violation from the Display Acknowledge error report
14	RW	0x0	ack_with_err_14 This bit retrieves the reserved(specific to device) from the Display Acknowledge error report
13	RW	0x0	ack_with_err_13 This bit retrieves the invalid transmission length from the Display Acknowledge error report
12	RW	0x0	ack_with_err_12 This bit retrieves the DSI VC ID Invalid from the Display Acknowledge error report

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x0	ack_with_err_11 This bit retrieves the not recognized DSI data type from the Display Acknowledge error report
10	RW	0x0	ack_with_err_10 This bit retrieves the checksum error(long packet only) from the Display Acknowledge error report
9	RW	0x0	ack_with_err_9 This bit retrieves the ECC error,multi-bit(detected and corrected) from the Display Acknowledge error report
8	RW	0x0	ack_with_err_8 This bit retrieves the ECC error,single-bit(detected and corrected) from the Display Acknowledge error report
7	RW	0x0	ack_with_err_7 This bit retrieves the reserved(specific to device) error from the Display Acknowledge error report
6	RW	0x0	ack_with_err_6 This bit retrieves the False Control error from the Display Acknowledge error report
5	RW	0x0	ack_with_err_5 This bit retrieves the HS Receive Timeout error from the Display Acknowledge error report
4	RW	0x0	ack_with_err_4 This bit retrieves the LP Transmit Sync error from the Display Acknowledge error report
3	RW	0x0	ack_with_err_3 This bit retrieves the Escape Mode Entry command error from the Display Acknowledge error report
2	RW	0x0	ack_with_err_2 This bit retrieves the EoT sync error from the Display Acknowledge error report
1	RW	0x0	ack_with_err_1 This bit retrieves the SoT Sync error from the Display Acknowledge error report
0	RW	0x0	ack_with_err_0 This bit retrieves the SoT error from the Display Acknowledge error report

**MIPIC\_MSK1**

Address: Operational Base + offset (0x000c8)

Masks the interrupt generation triggered by the ERROR\_ST1 reg

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12	RO	0x0	gen_pld_recv_err This bit indicates that during a generic interface packet read back, the payload FIFO becomes full and the received data is corrupted.
11	RO	0x0	gen_pld_rd_err This bit indicates that during a DCS read data, the payload FIFO becomes empty and the data sent to the interface is corrupted
10	RO	0x0	gen_pld_send_err This bit indicates that during a Generic interface packet build, the payload FIFO becomes empty and corrupt data is sent.
9	RO	0x0	gen_pld_wr_err This bit indicates that the system tried to write a payload data through the Generic interface and the FIFO is full. Therefore, the command is not written.
8	RO	0x0	gen_cmd_wr_err This bit indicates that the system tried to write a command through the Generic interface and the FIFO is full. Therefore, the command is not written.
7	RO	0x0	dpi_pld_wr_err This bit indicates that during a DPI pixel line storage, the payload FIFO becomes full and the data stored is corrupted.
6	RO	0x0	eotp_err This bit indicates that the EOTP packet is not received at the end of the incoming peripheral transmission.
5	RO	0x0	pkt_size_err This bit indicates that the packet size error is detected during the packet reception.
4	RO	0x0	crc_err This bit indicates that the CRC error is detected in a received packet.
3	RO	0x0	ecc_multi_err This bit indicates that the ECC multiple error is detected and corrected in a received packet.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RO	0x0	ecc_single_err This bit indicates that the ECC single error is detected and corrected in a received packet.
1	RO	0x0	to_lp_rx This bit indicates that the low-power reception timeout counter reached the end and contention detection is detected.
0	RO	0x0	to_hs_tx This bit indicates that the high-speed transmission timeout counter reached the end and contention detection is detected.

**MIPIC\_INT\_FORCE0**

Address: Operational Base + offset (0x000d8)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:21	RO	0x0	reserved
20	RO	0x0	dphy_errors_4 This bit indicates LP1 contention error ErrContentionLP1 from Lane 0
19	RO	0x0	dphy_errors_3 This bit indicates LP0 contention error ErrContentionLP0 from Lane 0
18	RO	0x0	dphy_errors_2 This bit indicates control error ErrControl from Lane 0
17	RO	0x0	dphy_errors_1 This bit indicates ErrSyncEsc low-power data transmission synchronization error from Lane 0
16	RO	0x0	dphy_errors_0 This bit indicates ErrEsc escape entry error from Lane 0
15	RO	0x0	ack_with_err_15 This bit retrieves the DSI protocol violation from the Display Acknowledge error report
14	RO	0x0	ack_with_err_14 This bit retrieves the reserved(specific to device) from the Display Acknowledge error report
13	RO	0x0	ack_with_err_13 This bit retrieves the invalid transmission length from the Display Acknowledge error report

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RO	0x0	ack_with_err_12 This bit retrieves the DSI VC ID Invalid from the Display Acknowledge error report
11	RO	0x0	ack_with_err_11 This bit retrieves the not recognized DSI data type from the Display Acknowledge error report
10	RO	0x0	ack_with_err_10 This bit retrieves the checksum error(long packet only) from the Display Acknowledge error report
9	RO	0x0	ack_with_err_9 This bit retrieves the ECC error,multi-bit(detected and corrected) from the Display Acknowledge error report
8	RO	0x0	ack_with_err_8 This bit retrieves the ECC error,single-bit(detected and corrected) from the Display Acknowledge error report
7	RO	0x0	ack_with_err_7 This bit retrieves the reserved(specific to device) error from the Display Acknowledge error report
6	RO	0x0	ack_with_err_6 This bit retrieves the False Control error from the Display Acknowledge error report
5	RO	0x0	ack_with_err_5 This bit retrieves the HS Receive Timeout error from the Display Acknowledge error report
4	RO	0x0	ack_with_err_4 This bit retrieves the LP Transmit Sync error from the Display Acknowledge error report
3	RO	0x0	ack_with_err_3 This bit retrieves the Escape Mode Entry command error from the Display Acknowledge error report
2	RO	0x0	ack_with_err_2 This bit retrieves the EoT sync error from the Display Acknowledge error report
1	RO	0x0	ack_with_err_1 This bit retrieves the SoT Sync error from the Display Acknowledge error report

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RO	0x0	ack_with_err_0 This bit retrieves the SoT error from the Display Acknowledge error report

**MIPIC\_INT\_FORCE1**

Address: Operational Base + offset (0x000dc)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12	RO	0x0	gen_pld_recv_err This bit indicates that during a generic interface packet read back, the payload FIFO becomes full and the received data is corrupted.
11	RO	0x0	gen_pld_rd_err This bit indicates that during a DCS read data, the payload FIFO becomes empty and the data sent to the interface is corrupted
10	RO	0x0	gen_pld_send_err This bit indicates that during a Generic interface packet build, the payload FIFO becomes empty and corrupt data is sent.
9	RO	0x0	gen_pld_wr_err This bit indicates that the system tried to write a payload data through the Generic interface and the FIFO is full. Therefore, the command is not written.
8	RO	0x0	gen_cmd_wr_err This bit indicates that the system tried to write a command through the Generic interface and the FIFO is full. Therefore, the command is not written.
7	RO	0x0	dpi_pld_wr_err This bit indicates that during a DPI pixel line storage, the payload FIFO becomes full and the data stored is corrupted.
6	RO	0x0	eotp_err This bit indicates that the EOTP packet is not received at the end of the incoming peripheral transmission.
5	RO	0x0	pkt_size_err This bit indicates that the packet size error is detected during the packet reception.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RO	0x0	crc_err This bit indicates that the CRC error is detected in a received packet.
3	RO	0x0	ecc_multi_err This bit indicates that the ECC multiple error is detected and corrected in a received packet.
2	RO	0x0	ecc_single_err This bit indicates that the ECC single error is detected and corrected in a received packet.
1	RO	0x0	to_lp_rx This bit indicates that the low-power reception timeout counter reached the end and contention detection is detected.
0	RO	0x0	to_hs_tx This bit indicates that the high-speed transmission timeout counter reached the end and contention detection is detected.

**MIPIC\_VID\_SHADOW\_CTRL**

Address: Operational Base + offset (0x00100)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved
16	RW	0x0	vid_shadow_pin_req When set to 1, the video request is done by external pin. In this mode, vid_shadow_req is ignored
15:9	RO	0x0	reserved
8	RW	0x0	vid_shadow_req When set to 1, the DPI registers are copied to the auxiliary registers. After copying, this bit is auto cleared.
7:1	RO	0x0	reserved
0	RW	0x0	vid_shadow_en When set to 1, DPI receives the active configuration from the auxiliary registers. When this bit is set along with the vid_shadow_req bit, the auxiliary registers are automatically updated.

**MIPIC\_Copy0\_DPI\_VCID**

Address: Operational Base + offset (0x0010c)

The DPI interface configuration.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved

Bit	Attr	Reset Value	Description
1:0	RW	0x0	dpi_vid This field configures the DPI virtual channel id that is indexed to the Video mode packets.

**MIPIC\_Copy0 DPI\_COLOR\_CODING**

Address: Operational Base + offset (0x00110)

Bit	Attr	Reset Value	Description
31:9	RO	0x0	reserved
8	RW	0x0	en18_loosely When set to 1, this bit enables 18 loosely packed pixel stream.
7:4	RO	0x0	reserved
3:0	RW	0x0	dpi_color_coding This field configures the DPI color coding as follows: 000:16bit configuration 1 001:16bit configuration 2 010:16bit configuration 3 011:18bit configuration 1 100:18bit configuration 2 101:24bit

**MIPIC\_Copy0 LP\_CMD\_TIM**

Address: Operational Base + offset (0x00118)

Low-power Command Timing Configuration Register.

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RW	0x00	outvact_lpcmd_time outside VACT region command time. This field configures the time available to transmit a command in low-power mode. The time value is expressed in a number of bytes format. The number of bytes represents the maximum size of a packet that can fit in a line during the VSA, VBP, and VFP region. This field must be configured with a value greater than or equal to four bytes to allow the transmission of the DCTRL commands such as shutdown and colorm in low-power mode.
15:8	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	RW	0x00	invact_lpcmd_time Inside VACT region command time. This field configures the time available to transmit a command in low-power mode. The time value is expressed in a number of bytes format. The number of bytes represents the maximum size of the packet that can fit a line during the VACT region.

**MIPIC\_Copy0 VID\_MODE\_CFG**

Address: Operational Base + offset (0x00138)

Video mode configuration.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	vpg_orientation This field indicates the color bar orientation as follows: 0:Vertical mode 1:Horizontal mode
23:21	RO	0x0	reserved
20	RW	0x0	vpg_mode This field is to select the pattern 0:Color bar(horizontal or vertical) 1:BER pattern(vertical only)
19:17	RO	0x0	reserved
16	RW	0x0	vpg_en When set to 1, this bit enables the video mode pattern generator
15	RW	0x0	lpcmden When set to 1, this bit enables the command transmission only in low-power mode
14	RW	0x0	frame_BTA_ack When set to 1, this bit enables the request for an acknowledge response at the end of a frame
13	RW	0x0	en_lp_hfp When set to 1, this bit enables the return to low-power inside the HFP period when timing allows.
12	RW	0x0	en_lp_hbp When set to 1, this bit enables the return to low-power inside the HBP period when timing allows.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11	RW	0x0	en_lp_vact When set to 1, this bit enables the return to low-power inside the VACT period when timing allows.
10	RW	0x0	en_lp_vfp When set to 1, this bit enables the return to low-power inside the VFP period when timing allows.
9	RW	0x0	en_lp_vbp When set to 1, this bit enables the return to low-power inside the VBP period when timing allows.
8	RW	0x0	en_lp_vsa When set to 1, this bit enables the return to low-power inside the VSA period when timing allows.
7:2	RO	0x0	reserved
1:0	RW	0x0	vid_mode_type This field indicates the video mode transmission type as follows: 00: Non-burst with sync pulses 01: Non-burst with sync events 10 and 11: Burst with sync pulses

**MIPIC\_Copy0 VID\_PKT\_SIZE**

Address: Operational Base + offset (0x0013c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13:0	RW	0x0000	vid_pkt_size This field configures the number of pixels on a single video packet. If you use the 18-bit mode and do not enable loosely packed stream, this value must be a multiple of 4.

**MIPIC\_Copy0 VID\_NUM\_CHUNKS**

Address: Operational Base + offset (0x00140)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12:0	RW	0x0000	num_chunks This field configures the number of chunks to be transmitted during a line period (a chunk is a video packet or a null packet)

**MIPIC\_Copy0 VID\_NULL\_SIZE**

Address: Operational Base + offset (0x00144)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12:0	RW	0x0000	null_pkt_size This field configures the number of bytes in a null packet

**MIPIC\_Copy0 VID\_HSA\_TIME**

Address: Operational Base + offset (0x00148)

Line timing configuration.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	hsa_time This field configures the Horizontal Synchronism Active period in lane byte clock cycles.

**MIPIC\_Copy0 VID\_HBP\_TIME**

Address: Operational Base + offset (0x0014c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	hbp_time This field configures the Horizontal Back Porch period in lane byte clock cycles

**MIPIC\_Copy0 VID\_HLINE\_TIME**

Address: Operational Base + offset (0x00150)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:15	RO	0x0	reserved
14:0	RW	0x0000	hline_time This field configures the size of the total lines counted in lane byte cycles.

**MIPIC\_Copy0 VID\_VSA\_LINES**

Address: Operational Base + offset (0x00154)

Vertical timing configuration.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9:0	RW	0x000	vsa_lines This field configures the Vertical Synchronism Active period measured in number of horizontal lines.

**MIPIC\_Copy0 VID\_VBP\_LINES**

Address: Operational Base + offset (0x00158)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9:0	RW	0x000	vbp_lines This field configures the Vertical Back Porch period measured in horizontal lines.

**MIPIC\_Copy0 VID\_VFP\_LINES**

Address: Operational Base + offset (0x0015c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9:0	RW	0x000	vfp_lines This field configures the Vertical Front Porch period measured in horizontal lines.

**MIPIC\_Copy0 VID\_VACTIVE\_LINES**

Address: Operational Base + offset (0x00160)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13:0	RW	0x0000	v_active_line This field configures the Vertical Active period measured in horizontal lines.

## 37.5 Application Notes

Low Power Mode is a special feature for D-PHY. You can control this function by using proper registers from the Innosilicon D-PHY with few operations. The following is a step by step instruction for low power mode in and out.

Perform the following steps to configure the DPI packet transmission:

Step1:Global configuration:

Configure n\_lanes (PHY\_IF\_CFG-[1:0]) to define the number of lanes in which the controller has to perform high-speed transmissions.

Step2:Configure the DPI Interface to define how the DPI interface interacts with the controller.

Configure dpi\_vid (DPI\_CFG-[1:0]): This field configures the virtual channel that the packet generated by the DPI interface is indexed to.

Configure dpi\_color\_coding (DPI\_CFG-[4:2]): This field configures the bits per pixels that the interface transmits and also the variant configuration of each bpp. If you select 18 bpp, and the Enable\_18\_loosely\_packed is not active, the number of pixels per line should be a multiple of four.

Configure dataen\_active\_low (DPI\_CFG-[5]): This bit configures the polarity of the dpidataen signal and enables if it is active low.

Configure vsync\_active\_low( DPI\_CFG-[6]): This bit configures the polarity of the dpivsync signal and enables if it is active low.

Configure vsync\_active\_low( DPI\_CFG-[7]): This bit configures the polarity of the dpivsync signal and enables if it is active low.

Configure vsync\_active\_low( DPI\_CFG-[8]): This bit configures the polarity of the dpishutdn signal and enables if it is active low.

Configure vsync\_active\_low( DPI\_CFG-[9]): This bit configures the polarity of the dpicolorlm signal and enables if it is active low.

Configure en18\_loosely( DPI\_CFG-[10]): This bit configures if the pixel packing is done loosely or packed when dpi\_color\_coding is 18 bpp. This bit enables loosely packing.

Step3: Select the Video Transmission Mode to define how the processor requires the video line to be transported through the DSI link.

Configure low-power transitions (VID\_MODE\_CFG-[8:3]): This defines the video line to be transported through the DSI link.

Configure low-power transitions (VID\_MODE\_CFG-[8:3]): This defines the video periods which are permitted to go to low-power if there is available time to do so.

Configure frame\_BTA\_ack (VID\_MODE\_CFG-[11]): This specifies if the controller should request the peripheral acknowledge message at the end of frames.

Burst mode: In this mode, the entire active pixel line is buffered into a FIFO and transmitted in a single packed with no interruptions. This transmission mode requires that the DPI Pixel FIFO has the capacity to store a full line of active pixel data inside it. This mode is optimally used if the difference between pixel required bandwidth and DSI link bandwidth is very different. This enables the DWC\_mipi\_dsi\_host to quickly dispatch the entire active video line in a single burst of data and then return to low-power mode.

Configure the register fiedl vid\_mode\_type (VID\_MODE\_CFG-[10]), num\_chunks (VID\_PKT\_CFG-[20:11]), and null\_pkt\_size (VID\_PKT\_CFG-[30:21]) are automatically ignored by the DWC\_mipi\_dsi\_host.

Non-Burst mode: In this mode, the processor uses the partitioning properties of the DWC\_mipi\_dsi\_host to divide the video line transmission into several DSI packets. This is done to match the pixel required bandwidth with the DSI link bandwidth. With this mode, the controller configuration does not require a full line of pixel data to be stored inside the DPI Pixel FIFO. It requires only the content of one video packet.

Configure the vid\_mode\_type field (VID\_MODE\_CFG-[2:1]) with 2'b0x.

Configure the vid\_mode\_type field (VID\_MODE\_CFG-[2:1]) with 2'b00x to enable the transmission of sync pulses.

Configure the vid\_mode\_type field (VID\_MODE\_CFG-[2:1]) with 2'b01 to enable the transmission of sync events.

Configure the vid\_mode\_type field (VID\_MODE\_CFG-[10:0]) with the number of pixels to be

transmitted in a single packet.

Configure the en\_multi\_pkt field (VID\_MODE\_CFG-[9]) to enable the division of the active video transmission into more than one packet.

Configure the num\_chunks field (VID\_MODE\_CFG-[20:11]) with the number of video chunks that the active video transmission is divided into.

Configure the en\_null\_pkt field (VID\_MODE\_CFG-[10]) to enable the insertion of null packets between video packets.

The field is effective only when en\_multi\_pkt field is activated, otherwise the controller ignores it and does not sent the null packets.

Configure the null\_pkt\_size field (VID\_MODE\_CFG-[30:21]) with the actual size of the inserted null packet.

Step4: Define the DPI Horizontal timing configuration as follows:

Configure the hline\_time field (TMR\_LINE\_CFG-[31:18]) with the time taken by a DPI video line accounted in Clock Lane bytes clock cycles (for a clock lane at 500 MHz the Lane byte clock period is 8 ns). When the DPI clock and Clock Lane clock are not multiples, the hline\_time is a result of a round of a number. If the DWC\_mipi\_dsi\_host is configured to go to low-power, it is possible that the error included in a line is incremented with the next one. At the end of several lines, the DWC\_mipi\_dsi\_host can have a number of errors that can cause a malfunction of the video transmission.

Configure the hsa\_time field (TMR\_LINE\_CFG-[8:0]) with the time taken by a DPI Horizontal Sync Active period accounted in Clock Lane byte clock cycles (normally a period of 8ns).

Configure the hbp\_time field (TMR\_LINE\_CFG-[17:9]) with the time taken by a DPI Horizontal Sync Active period accounted in Clock Lane byte clock cycles (normally a period of 8ns). Special attention should be given to the calculation of this parameter.

Step5: Define the Vertical line configuration:

Configure the vsa\_lines field (VTIMING\_CFG-[3:0]) with the number of lines existing in the DPI Vertical Sync Active period.

Configure the vbp\_lines field (VTIMING\_CFG-[9:4]) with the number of lines existing in the DPI Vertical Back Porch period.

Configure the vfp\_lines field (VTIMING\_CFG-[15:10]) with the number of lines existing in the DPI Vertical Front Porch period.

Configure the v\_active\_lines field (VTIMING\_CFG-[26:16]) with the number of lines existing in the DPI Vertical Active period.

## Chapter 38 Global Positioning System (GPS)

### 38.1 Overview

The GPS is a high-performance baseband device which has an AHB master interface and an AHB slave interface.

CPU can access GPS registers through the AHB slave interface.

The GPS has a 32-channel DMA inside, which can read/write data to system memory through the AHB master interface.

The GPS supports following features:

- Single chip, integrate GPS baseband with CPU
- 32 DMA channels for AHB master access
- Complete L1-band, C/A, and NMEA-0183 compatibility
- Support reference frequency 16.368MHz
- High sensitivity for indoor fixes
- Low power consumption
- Low cost with smaller size
- Multi modes support both standalone GPS and A\_GPS

### 38.2 Block Diagram

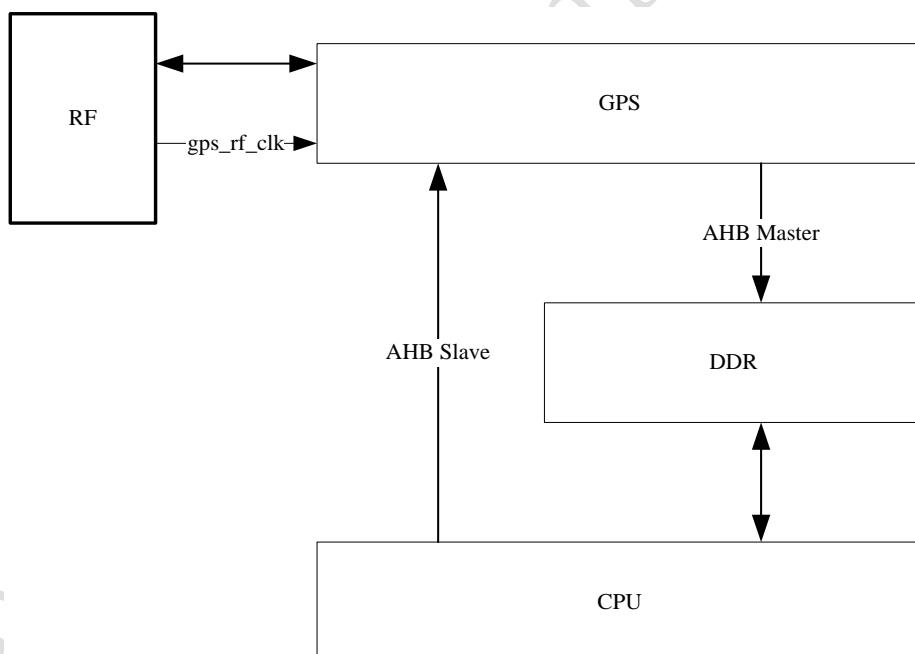


Fig. 38-1 RK3288 GPS block diagram

As shown in Fig. 39-1, the GPS controller should be connected to RF chip through GPIO. The RF chip has 3 1-bit signals output to GPS controller inside chip. They are GPS\_RF\_CLK, GPS\_SIG, GPS\_MAG.

The CPU can start RF chip through GPIO, after GPS configuring completion.

### 38.3 Register Summary

The base address of GPS is 0xff000000.

### 38.3.1 Base band register summary

Table 39-1 RK3288 base band register summary

Name	Offset	Size	Reset Value	Description
BB_CTRL	0x400	W	0x0	base band control register
BB_START_ADR	0x404	W	0x0	correlator start address register
BB_DS_PARAMETER	0x408	W	0x0	down-sample control register
GPS_INT_ENA	0x40c	W	0x0	interrupt enable register
GPS_INT_STATUS	0x410	W	0x0	interrupt status register
BB_CHN_COR_STATUS	0x414	W	0x0	channel correlator status register
BB_CHN_COR_VALID	0x418	W	0x0	channel correlator valid register
BB_RF_TIMER_VAL	0x41c	W	0x0	RF clock timer value register
BB_RF_WT_ADDR	0x420	W	0x1fffffff	RF FIFO write address register
CH_MISCx	x<<5+0	W	N/A	control information
CH_CAR_NCOx	x<<5+4	W	N/A	carrier NCO
CH_CODE_NCOx	x<<5+8	W	N/A	code NCO
CH_DMA_ADDRx	x<<5+12	W	N/A	dma address
CH_CAR_FREQx	x<<5+16	W	N/A	carrier frequency
CH_CODE_FREQx	x<<5+20	W	N/A	carrier frequency

Note: x is for the channel number from 0-31

### 38.3.2 ACC operation register summary

Table 39-2 RK3288 GPS acc register summary

Name	Offset	Size	Reset Value	Description
ACC_CTRL	0x8000	W	0x0	acc control register
DMA_CH0_START_ADDR	0x8004	W	0xffffffff	channel 0 dma start address
DMA_CH1_START_ADDR	0x800c	W	0xaaaaaaaa	channel 1 dma start address
DMA_CH2_START_ADDR	0x8014	W	0x55555555	channel 2 dma start address
FFT_VLD_NUM_START_POINT	0x801c	W	0x0	FFT_START_POINT, FFT_VALID_NUM
FFT_COR_NUM_LOOP_NUM	0x8020	W	0x0	FFT_LOOP_NUM, TOT_EPD_COR_NUM, EPD_START_COR_NUM
EPD_MAXCORID	0x8024	W	0x0	EPD_MAXCORID, MAX_FREQINDEX
EPD_MAX_DATA_MAN	0x8028	W	0x0	the maximum energy mantissa
EPD_MAX_DATA_EXP	0x802c	W	0x0	the exponent of the maximum energy
EPD_MAX_RIGHT_MAN	0x8030	W	0x0	the mantissa of the right data to the max energy
EPD_MAX_RIGHT_EXP	0x8034	W	0x0	the exponent of the right data to the max energy
EPD_MAX_LEFT_MAN	0x8038	W	0x0	the mantissa of the left data to the max energy

Name	Offset	Size	Reset Value	Description
EPD_MAX_LEFT_EXP	0x803c	W	0x0	the exponent of the left data to the max energy
EPD_L0_DATA_MANT	0x8040	W	0x0	the mantissa of the max energy of the cor0 data
EPD_L0_DATA_EXP	0x8044	W	0x0	the exponent of the max energy of the cor0 data
EPD_L1_DATA_MANT	0x8048	W	0x0	the mantissa of the max energy of the cor1 data
EPD_L1_DATA_EXP	0x804c	W	0x0	the exponent of the max energy of the cor1 data
EPD_L2_DATA_MANT	0x8050	W	0x0	the mantissa of the max energy of the cor2 data
EPD_L2_DATA_EXP	0x8054	W	0x0	the exponent of the max energy of the cor2 data
EPD_L3_DATA_MANT	0x8058	W	0x0	the mantissa of the max energy of the cor3 data
EPD_L3_DATA_EXP	0x805c	W	0x0	the exponent of the max energy of the cor3 data
EPD_L4_DATA_MANT	0x8060	W	0x0	the mantissa of the max energy of the cor4 data
EPD_L4_DATA_EXP	0x8064	W	0x0	the exponent of the max energy of the cor4 data
EPD_L5_DATA_MANT	0x8068	W	0x0	the mantissa of the max energy of the cor5 data
EPD_L5_DATA_EXP	0x806c	W	0x0	the exponent of the max energy of the cor5 data
EPD_L6_DATA_MANT	0x8070	W	0x0	the mantissa of the max energy of the cor6 data
EPD_L6_DATA_EXP	0x8074	W	0x0	the exponent of the max energy of the cor6 data
EPD_L7_DATA_MANT	0x8078	W	0x0	the mantissa of the max energy of the cor7 data
EPD_L7_DATA_EXP	0x807c	W	0x0	the exponent of the max energy of the cor7 data
EPD_L8_DATA_MANT	0x8080	W	0x0	the mantissa of the max energy of the cor8 data
EPD_L8_DATA_EXP	0x8084	W	0x0	the exponent of the max energy of the cor8 data
EPD_L9_DATA_MANT	0x8088	W	0x0	the mantissa of the max energy of the cor9 data
EPD_L9_DATA_EXP	0x808c	W	0x0	the exponent of the max energy of the cor9 data

## 38.4 Interface Description

The IOMUX configuration of GPS is shown below:

Table 39-3 RK3288 GPS IOMUX Setting

Module Pin	IO	Pad Name	IOMUX Setting
gps_rfclk	I	IO_UART3GPScts_GPSrfclk_GPST1clk_GPIO30g_pio7b1	GPIO7B_IOMUX[3:2]=2'b10
gps_mag	I	IO_UART3GPSSin_GPSmag_HSADCT1data0_GPI_O30gpio7a7	GPIO7A_IOMUX[15:14]=2'b10
gps_sig	I	IO_UART3GPSSout_GPSsig_HSADCT1data1_GPI_O30gpio7b0	GPIO7B_IOMUX[1:0]=2'b10

Notes: 1. *I*=input, *O*=output, *I/O*=input/output, *bidirectional*

## **38.5 Application note**

GPS has 2 interrupt output signals: gps\_irq and gps\_timer\_irq. The gps\_irq interrupt has a system interrupt ID 61, and gps\_timer\_irq has a system interrupt ID 62.

## Chapter 39 TSP(Transport Stream Processing Module)

### 39.1 Overview

The Transport Stream Processing Module(TSP) is designed for processing Transport Stream Packets, including receiving TS packets, PID filtering, TS descrambling, De-multiplexing and TS outputting. Processed data are transferred to memory buffer which are continued to be processing by software.

TPS supports the following features:

- Supports two TS input channels and one TS output channel
- Supports 4 TS Input Mode: sync/valid mode in the case of serial TS input; nosync/valid mode, sync/valid, sync/burst mode in the case of parallel TS input
- Supports serial and parallel output mode with PCR adjustment, and lsb-msb or msb-lsb bit ordering can be chosen in the serial output mode
- Supports 2 TS sources: demodulators and local memory
- Supports 2 Built-in PTIs(Programmable Transport Interface) to process TS simultaneously
- Supports 1 PVR(Personal Video Recording) output channel
- 1 built-in multi-channel DMA Controller
- DMAC supports:
  - Word alignment transfer
  - Fixed and incrementing addressing
  - Word size transfer
  - burst modes: Incr4, Incr8, Inc16; burst transfer will be done with INCR mode if the remaining data or address space is not capable to perform a complete burst transfer
  - Hardware/software trigger mode
  - LLP(List Link Programming) Mode
  - DMA done and error interrupt for each PTI channel
- Each PTI supports
  - 64 PID filters
  - TS descrambling with 16 sets of Control Word under CSA v2.0 standard, up to 104Mbps
  - 16 PES/ES filters with PTS/DTS extraction and ES start code detection
  - 4/8 PCR extraction channels
  - 64 Section filters with CRC check, and three interrupt mode: stop per unit, full-stop, recycle mode with version number check
  - PID done and error interrupts for each channel
  - PCR/DTS/PTS extraction interrupt for each channel

### 39.2 Block Diagram

The TSP comprises of following components:

- AMBA AHB slave interface
- Register block
- PTI
- DMAC
- TS Out Interface

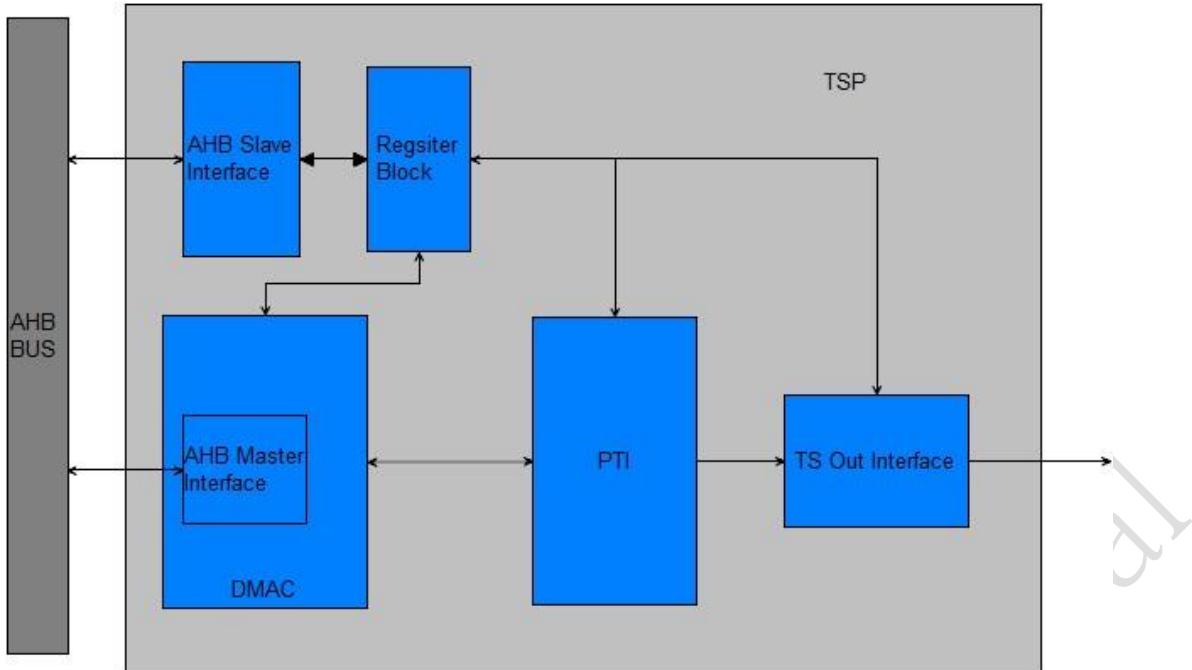


Fig. 39-1 TSP architecture

### AHB Slave INTERFACE

The host processor can get access to the register block through AHB slave interface. The slave interface supports 32bit access.

### Register block

All registers in the TSP are addressed at 32-bit boundaries to remain consistent with the AHB bus. Where the physical size of any register is less than 32-bits wide, the upper unused bits of the 32-bit boundary are reserved. Writing to these bits has no effect; reading from these bits returns 0.

### PTI

Most of the TS processing are dealt with PTI. TS packets are re-synchronized, filtered, descrambled and demultiplexing, and the processed packets are transferred to memory buffer to be processed further by software. The embedded TS in interface can receive TS packets by connecting to a compliant TS demodulator. TS stream stored in the local memory is another source to feed into PTI through by using LLP DMA mode.

### TS Out Interface

TS out interface can output either PID-filtered or non-PID-filtered TS packets from one PTI channel in a certain stream mode as configured. The TS receiver conforms to the stream mode to receive the TS packets.

### DMAC

The DMAC performs all DMA transfers which get access to memory.

## 39.3 Function Description

### 39.3.1 TS Stream of TS\_IN Interface

TS\_IN interface supports 4 input TS stream mode: sync/valid serial mode, sync/valid parallel mode, sync/burst parallel mode, nosync/valid parallel mode.

### A. Sync/Valid Serial Mode

In this mode, TS\_IN interface takes use of TSI\_SYNC and TSI\_VALID clocked with TSI\_CLK signal to sample input serial TS packet data.

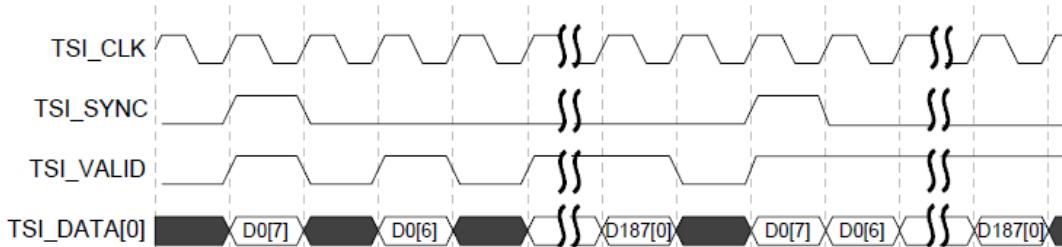


Fig. 39-2 Sync/Valid Serial Mode with Msb-Lsb Bit Ordering

TSI\_SYNC must be active high together with TSI\_VALID when indicating the first valid bit of a TS packet, and TSI\_VALID indicates the 188\*8 valid bits of a TS packet. TSI supports both msb-lsb and lsb-msb bit ordering.

### B. Sync/Valid Parallel Mode

In this mode, TS\_IN interface takes use of TSI\_SYNC and TSI\_VALID clocked with TSI\_CLK signal to sample input parallel TS packet data.

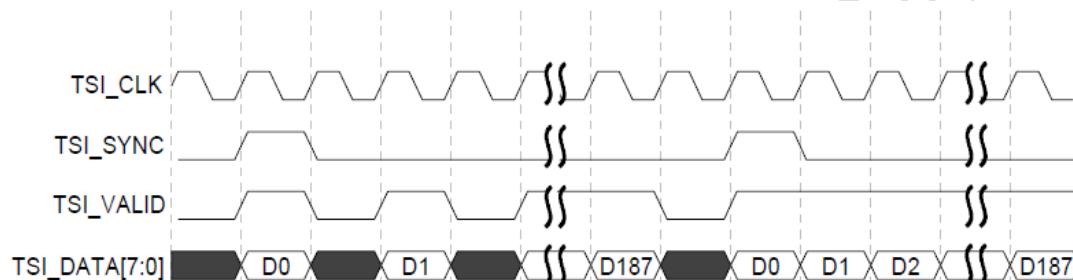


Fig. 39-3 Sync/valid Parallel Mode

TSI\_SYNC must be active high together with TSI\_VALID when indicating the first valid byte of a TS packet, and TSI\_VALID indicates the 188 valid byte of a TS packet.

### C. Sync/Burst Parallel Mode

In this mode, TSI only takes use of TSI\_SYNC to sample input parallel TS packet data.

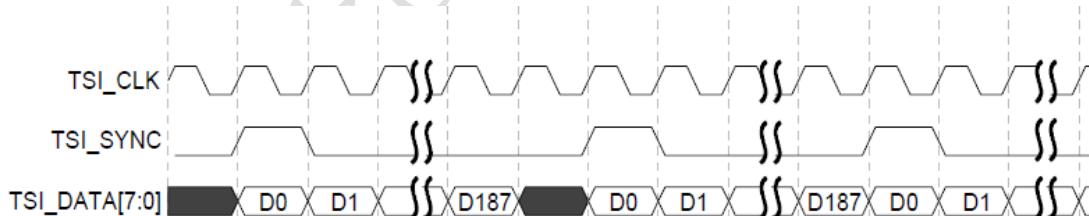


Fig. 39-4 Sync/Burst Parallel Mode

When active high, TSI\_SYNC implies the first valid byte of a TS packet and remaining 187 valid bytes of a TS packet are upcoming within the following successive 187 clock cycles.

### D. Nosync/Valid Parallel Mode

In this mode, TSI only takes uses of TSI\_VALID to sample input parallel TS packet data.

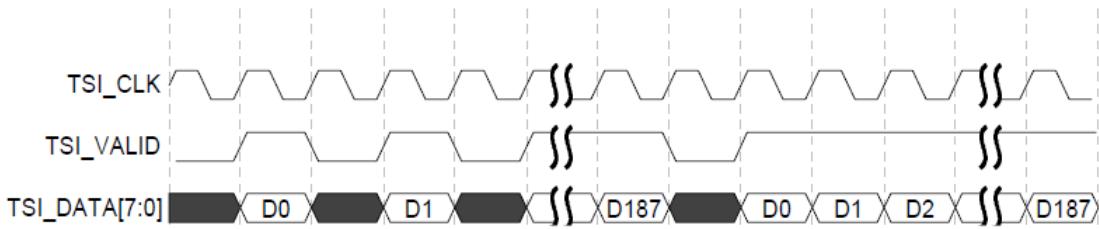


Fig. 39-5 Nosync/Valid Parallel Mode

When active high, TSI\_VALID implies a valid byte of a TS packet.

### 39.3.2 TS output of TS Out Interface

TS out interface transmit the TS data in two mode: serial mode and parallel mode. In the serial mode, the bit order can be lsb-msb or msb-lsb.

The TS\_SYNC will be active high when indicating the header of the TS packets, and it only lasts for one cycle. TS\_VALID will be active high when the output TS data is valid. The output data is 188 byte TS packet data.

TS out interface also stamp the TS output stream with new PCR value, making PCR adjustment. PCR is used to measure the transport rate.

$$PCR(i) = PCR\_base(i) \times 300 + PCR\_ext(i)$$

where:

$$PCR\_base(i) = ((system\_clock\_frequency \times t(i)) DIV 300) \% 2^{33}$$

$$PCR\_ext(i) = ((system\_clock\_frequency \times t(i)) DIV 1) \% 300$$

$$transport\_rate(i) = \frac{((i' - i'') \times system\_clock\_frequency)}{PCR(i') - PCR(i'')}$$

Where

$i'$  is the index of the byte containing the last bit of the immediately following program\_clock\_reference\_base field applicable to the program being decoded.

$i$  is the index of any byte in the Transport Stream for  $i'' < i < i'$ .

$i''$  is the index of the byte containing the last bit of the most recent program\_clock\_reference\_base field applicable to the program being decoded.

System clock is 27Mhz.

### 39.3.3 Demux and descrambling

Each PTI has 64 PID channels to deal with demultiplexing and descrambling operation.

The PTI can descramble the TS Packets which are scrambled with CSA v2.0 standard. The TS packets can be scrambled either in TS level or PES level.

The demux module can do the section filtering, pes filtering and es filtering, or directly output TS packets.

## 39.4 Register Description

### 39.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
TSP_GCFG	0x0000	W	0x00000000	Global Configuration Register
TSP_PVR_CTRL	0x0004	W	0x00000000	PVR Control Register
TSP_PVR_LEN	0x0008	W	0x00000000	PVR DMA Transaction Length
TSP_PVR_ADDR	0x000c	W	0x00000000	PVR DMA transaction starting address
TSP_PVR_INT_STS	0x0010	W	0x00000000	PVR DMA Interrupt Status Register
TSP_PVR_INT_ENA	0x0014	W	0x00000000	DMA Interrupt Enable Register
TSP_TSOUT_CTRL	0x0018	W	0x00000000	TS Out Control Register
TSP_PTIx_CTRL	0x0100	W	0x00000000	PTI Channel Control Register
TSP_PTIx_LLPCFG	0x0104	W	0x00000000	LLP DMA Control Register
TSP_PTIx_LLPCBASE	0x0108	W	0x00000000	LLP Descriptor BASE Address
TSP_PTIx_LLPCWRIT	0x010c	W	0x00000000	LLP DMA Writing Software Descriptor Counter
TSP_PTIx_LLPCREAD	0x0110	W	0x00000000	LLP DMA Reading Hardware Descriptor Counter
TSP_PTIx_PID_STS0	0x0114	W	0x00000000	PTI PID Channel Status 0 Register
TSP_PTIx_PID_STS1	0x0118	W	0x00000000	PTI PID Channel Status 1 Register
TSP_PTIx_PID_STS2	0x011c	W	0x00000000	PTI PID Channel Status 2 Register
TSP_PTIx_PID_STS3	0x0120	W	0x00000000	PTI PID Channel Status 3 Register
TSP_PTIx_PID_INT_ENA0	0x0124	W	0x00000000	PID Interrupt Enable Register 0
TSP_PTIx_PID_INT_ENA1	0x0128	W	0x00000000	PID Interrupt Enable Register 1
TSP_PTIx_PID_INT_ENA2	0x012c	W	0x00000000	PID Interrupt Enable Register 2
TSP_PTIx_PID_INT_ENA3	0x0130	W	0x00000000	PID Interrupt Enable Register 3
TSP_PTIx_PCR_INT_STS	0x0134	W	0x00000000	PTI PCR Interrupt Status Register
TSP_PTIx_PCR_INT_ENA	0x0138	W	0x00000000	PTI PCR Interrupt Enable Register
TSP_PTIx_PCRn_CTRL	0x013c	W	0x00000000	PID PCR Control Register

Name	Offset	Size	Reset Value	Description
TSP_PTIx_PCRn_H	0x015c	W	0x00000000	High Order PCR value
TSP_PTIx_PCRn_L	0x0160	W	0x00000000	Low Order PCR value
TSP_PTIx_DMA_STS	0x019c	W	0x00000000	LLP DMA Interrupt Status Register
TSP_PTIx_DMA_ENA	0x01a0	W	0x00000000	DMA Interrupt Enable Register
TSP_PTIx_DATA_FLA_G0	0x01a4	W	0x00000000	PTI_PID_WRITE Flag 0
TSP_PTIx_DATA_FLA_G1	0x01a8	W	0x00000000	PTI_PID_WRITE Flag 1
TSP_PTIx_LIST_FLAG	0x01ac	W	0x00000000	PTIx_LIST_WRITE Flag
TSP_PTIx_DST_STS0	0x01b0	W	0x00000000	PTI Destination Status Register
TSP_PTIx_DST_STS1	0x01b4	W	0x00000000	PTI Destination Status Register
TSP_PTIx_DST_ENA_0	0x01b8	W	0x00000000	PTI Destination Interrupt Enable Register
TSP_PTIx_DST_ENA_1	0x01bc	W	0x00000000	PTI Destination Interrupt Enable Register
TSP_PTIx_ECWn_H	0x0200	W	0x00000000	The Even Control Word High Order
TSP_PTIx_ECWn_L	0x0204	W	0x00000000	The Even Control Word Low Order
TSP_PTIx_OCWn_H	0x0208	W	0x00000000	The Odd Control Word High Order
TSP_PTIx_OCWn_L	0x020c	W	0x00000000	The Odd Control Word Low Order
TSP_PTIx_PIDn_CTR_L	0x0300	W	0x00000000	PID Channel Control Register
TSP_PTIx_PIDn_BAS_E	0x0400	W	0x00000000	PTI Data Memory Buffer Base Address
TSP_PTIx_PIDn_TOP	0x0404	W	0x00000000	PTI Data Memory Buffer Top Address
TSP_PTIx_PIDn_WRITE	0x0408	W	0x00000000	PTI Data Memory Buffer Hardware Writing Address
TSP_PTIx_PIDn_READ	0x040c	W	0x00000000	PTI Data Memory Buffer Software Reading Address
TSP_PTIx_LISTn_BASE	0x0800	W	0x00000000	PTI List Memory Buffer Base Address
TSP_PTIx_LISTn_TOP	0x0804	W	0x00000000	PTI List Memory Buffer Top Address
TSP_PTIx_LISTn_WRITE	0x0808	W	0x00000000	PTI List Memory Buffer Hardware Writing Address

Name	Offset	Size	Reset Value	Description
TSP_PTIX_LISTn_READ	0x080c	W	0x00000000	PTI List Memory Buffer Software Reading Address
TSP_PTIX_PIDn_CFG	0x0900	W	0x00000008	PID Demux Configure Register
TSP_PTIX_PIDn_FILTER_0	0x0904	W	0x00000000	Fliter Word 0
TSP_PTIX_PIDn_FILTER_1	0x0908	W	0x00000000	Fliter Word 1
TSP_PTIX_PIDn_FILTER_2	0x090c	W	0x00000000	Fliter Word 2
TSP_PTIX_PIDn_FILTER_3	0x0910	W	0x00000000	Fliter Word 3

Notes: **Size** : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

### 39.4.2 Detail Register Description

#### TSP\_GCFG

Address: Operational Base + offset (0x0000)

Global Configuration Register

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6:4	RW	0x0	arbit_cnt DMA channel arbiter counter This field is used to adjust the priority of DMA channels to prevent one channel holds the highest priority for a long time. The 3-bit field sets the largest times for a DMA channel to hold the highest priority to send the bus request. After requested times reach this limit, the highest priority is passed to next DMA channel in order.
3	RW	0x0	tsout_on TS Output Module Switch 1: TS output module switched on 0: TS output module switched off
2	RW	0x0	pvr_on PVR Module Switch 1: PVR function turned on ; 0: PVR function turned off ;
1	RW	0x0	pti1_on PTI0 channel switch 1: PTI1 channel switched on 0: PTI1 channel switched off

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	pti0_on PTI0 channel switch 1: PTI0 channel switched on 0: PTI1 channel switched off

**TSP\_PVR\_CTRL**

Address: Operational Base + offset (0x0004)

PVR Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6	RW	0x0	fixaddr_en Fix Address Mode Select 1: fixed address mode; 0: incrementing address mode;
5:4	RW	0x0	burst_mode PVR burst mode PVR DMA burst mode 2'b00: INCR4 2'b01: INCR8 2'b10: INCR16 2'b11: Reserved
3:2	RW	0x0	source PVR Source Select TS source for PVR output. 00: non-PID-filtered TS packets in PTI0; 01: PID filtered TS packets in PTI0; 10: non-PID-filtered TS packets in PTI1; 11: PID-filtered TS packets in PTI1;
1	RWSC	0x0	stop PVR stop Write 1 to stop DMA channel. DMA will complete current burst transfer and then stop. It may take several cycles. 1: PVR Stop ; 0: no effect ;
0	RWSC	0x0	start PVR start Write 1 to start PVR. This bit will be cleared if PVR is stopped or PVR transaction is completed. 1: start PVR 0: no effect.

**TSP\_PVR\_LEN**

Address: Operational Base + offset (0x0008)

PVR DMA Transaction Length

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	len Transaction Length Transaction Length

**TSP\_PVR\_ADDR**

Address: Operational Base + offset (0x000c)

PVR DMA transaction starting address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	addr PVR DMA transaction starting address PVR DMA transaction starting address

**TSP\_PVR\_INT\_STS**

Address: Operational Base + offset (0x0010)

PVR DMA Interrupt Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	W1C	0x0	pvr_error PVR DMA transaction error 1: error response during PVR DMA transaction; 0: no error response during PVR DMA transaction;
0	W1C	0x0	pvr_done PVR DMA transaction done 1: PVR DMA transaction completed; 0: PVR DMA transaction not completed;

**TSP\_PVR\_INT\_ENA**

Address: Operational Base + offset (0x0014)

DMA Interrupt Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	pvr_error_ena PVR DMA Transaction Error Interrupt Enable 1: Error Interrupt Enabled 0: Error Interrupt Disabled
0	RW	0x0	pvr_done_ena PVR DMA Transaction Done Interrupt Enable 1: Done Interrupt Enabled 0: Done Interrupt Disabled

**TSP\_TSOUT\_CTRL**

Address: Operational Base + offset (0x0018)

TS Out Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6	RW	0x0	tso_sdo_sel TS serial data output 1: bit[0] use as serial data output ; 0: bit[7] use as serial data output ;
5	RW	0x0	tso_clk_phase TS output clock phase 0: ts output clock; 1: inverse of ts output clock.
4	RW	0x0	mode TS Output mode Selection Output mode select: 0: Serial Mode 1: Parallel Mode
3	RW	0x0	bit_order ts output serial data byte order Indicates that the output serial data byte order, ignored in the parallel: 0: MSB to LSB 1: LSB to MSB
2:1	RW	0x0	source TS Output Source Select TS source for TS out. 00: non-PID-filtered TS packets in PTI0; 01: PID filtered TS packets in PTI0; 10: non-PID-filtered TS packets in PTI1; 11: PID-filtered TS packets in PTI1;
0	RW	0x0	start TS out start 1: to start TS out function ; 0: to stop TS out function;

**TSP\_PTIx\_CTRL**

Address: Operational Base + offset (0x0100)

PTI Channel Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RO	0x0	reserved
21	RW	0x0	tsi_sdi_sel TS Serial Data Input Select 1: bit[0] use as serial input data 0: bit[7] use as serial input data

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
20:19	RW	0x0	tsi_error_handle TS ERROR Handle 00: don't output 01: set the error indicator to 1 10: don't care
18	RW	0x0	clk_phase_sel ts input clock phase select 1'b0: ts input clock 1'b1: inverse of ts input clock
17:16	RW	0x0	demux_burst_mode Demux DMA Burst Mode Demux DMA Mode 2'b00: INCR4 2'b01: INCR8 2'b10: INCR16 2'b11: Reserved
15	RW	0x0	sync_bypass Bypass mode Selection 1'b1: Bypass mode, indicating that input TS packets will not be resynchronized and directly fed into the following modules; 1'b0: Synchronous mode, default, indicating that input TS packets will be resynchronized;
14	RW	0x0	cw_byteorder Control Word format Configuration 0: Default: first byte of the word is the highest byte 1: first byte of the word is the lowest byte
13	RW	0x0	cm_on CSA Conformance Mechanism Configuration CSA Conformance Mechanism 0: CM turned off 1: CM turned on
12:11	RW	0x0	tsi_mode TSI Input Mode Selection Input mode selection: 00: Serial Sync/valid Mode 01: Parallel Sync/valid Mode 10: Parallel Sync/burst Mode 11: Parallel Nosync/valid Mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
10	RW	0x0	tsi_bit_order input serial data order Indicates that the input serial data byte order, ignored in the parallel mode: 0: MSB to LSB 1: LSB to MSB
9	RW	0x0	tsi_sel TS Input Source Select Select input TS source 1'b1: HSADC ; 1'b0: internal memory ;
8	RW	0x0	out_byteswap Output byteswap function When enabled, the word to be transferred to memory buffer "B4B3B2B1" is performed byteswapping to "B1B2B3B4".
7	RW	0x0	in_byteswap Input TS Word Byteswap When enabled, the input TS word "B4B3B2B1" is performed byteswapping to "B1B2B3B4".
6:4	RW	0x0	unsync_times TS Header Unsynchronized Times If synchronous mode is selected. This field sets the successive times of TS packet header error to re-lock TS header when TS is in locked status;
3:1	RW	0x0	sync_times TS Header Synchronized Times If synchronous mode is selected. This field sets the successive times of finding TS packet header to lock the TS header when TS is in unlocked status;
0	RWSC	0x0	clear Software clear signal It will reset the core register . It will take several cycles. After reset done, soft_reset will be low. 1. reset; 0. no effect.

**TSP\_PTIX\_LLPCFG**

Address: Operational Base + offset (0x0104)

LLP DMA Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9:8	RW	0x0	<p>threshold LLP Transfer Threshold The depth for LLP descriptors is 64. An interrupt will be asserted when transfer reaches the threshold set if DMA transfer interrupt is enabled.</p> <p>00: 1/1 depth 01: 1/2 depth 10: 1/4 depth 11: 1/8 depth</p>
7:6	RW	0x0	<p>burst_mode LLP DMA Burst Mode LLP DMA Burst Mode 2'b00: INCR4 2'b01: INCR8 2'b10: INCR16 2'b11: Reserverd</p>
5	RW	0x0	<p>hw_trigger Hardware Trigger Select 1. hardware trigger; 0. software trigger;</p>
4	RW	0x0	<p>fix_addr_en Fix Address Mode Select 1: fixed address mode; 0: incrementing address mode;</p>
3	W1C	0x0	<p>cfg_done LLP DMA Configuration Done When all descriptors of LLP are configured, write 1 to to this bit. The core will clear this bit when llp transction is finished ;</p>
2	RW	0x0	<p>pause LLP DMA Pause Write 1 to Pause DMA channel . DMA will complete current burst transfer and then pause. All register stay unchange. If software write 0 later , It will continue to work. It may take several cycles to pause. 1: pause; 0: continue to work ;</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	W1C	0x0	stop LLP DMA Stop Write 1 to stop DMA channel. DMA will complete current burst transfer and then stop. It may takes several cycles. 1: stop ; 0: no effect ;
0	W1C	0x0	start LLP DMA start Write 1 to start DMA Channel , self clear after 1 cycle. 1: start ; 0: no effect

**TSP\_PTIX\_LL\_P\_BASE**

Address: Operational Base + offset (0x0108)

LLP Descriptor BASE Address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	addr LLP Descriptor BASE Address LLP Descriptor BASE address

**TSP\_PTIX\_LL\_P\_WRITE**

Address: Operational Base + offset (0x010c)

LLP DMA Writing Software Descriptor Counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x00	counter LLP DMA Writing Software Descriptor Counter LLP DMA Writing Software Descriptor Counter

**TSP\_PTIX\_LL\_P\_READ**

Address: Operational Base + offset (0x0110)

LLP DMA Reading Hardware Descriptor Counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RO	0x00	counter LLP DMA Reading Hardware Descriptor Counter Counter LLP DMA Reading Hardware Descriptor Counter Counter

**TSP\_PTIX\_PID\_STS0**

Address: Operational Base + offset (0x0114)

PTI PID Channel Status 0 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	pid31_done PID31 Channel Status 1 means done
30	W1C	0x0	pid30_done PID30 Channel Status 1 means done
29	W1C	0x0	pid29_done PID29 Channel Status 1 means done
28	W1C	0x0	pid28_done PID28 Channel Status 1 means done
27	W1C	0x0	pid27_done PID27 Channel Status 1 means done
26	W1C	0x0	pid26_done PID26 Channel Status 1 means done
25	W1C	0x0	pid25_done PID25 Channel Status 1 means done
24	W1C	0x0	pid24_done PID24 Channel Status 1 means done
23	W1C	0x0	pid23_done PID23 Channel Status 1 means done
22	W1C	0x0	pid22_done PID22 Channel Status 1 means done
21	W1C	0x0	pid21_done PID21 Channel Status 1 means done
20	W1C	0x0	pid20_done PID20 Channel Status 1 means done
19	W1C	0x0	pid19_done PID19 Channel Status 1 means done
18	W1C	0x0	pid18_done PID18 Channel Status 1 means done
17	W1C	0x0	pid17_done PID17 Channel Status 1 means done

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16	W1C	0x0	pid16_done PID16 Channel Status 1 means done
15	W1C	0x0	pid15_done PID15 Channel Status 1 means done
14	W1C	0x0	pid14_done PID14 Channel Status 1 means done
13	W1C	0x0	pid13_done PID13 Channel Status 1 means done
12	W1C	0x0	pid12_done PID12 Channel Status 1 means done
11	W1C	0x0	pid11_done PID11 Channel Status 1 means done
10	W1C	0x0	pid10_done PID10 Channel Status 1 means done
9	W1C	0x0	pid9_done PID9 Channel Status 1 means done
8	W1C	0x0	pid8_done PID8 Channel Status 1 means done
7	W1C	0x0	pid7_done PID7 Channel Status 1 means done
6	W1C	0x0	pid6_done PID6 Channel Status 1 means done
5	W1C	0x0	pid5_done PID5 Channel Status 1 means done
4	W1C	0x0	pid4_done PID4 Channel Status 1 means done
3	W1C	0x0	pid3_done PID3 Channel Status 1 means done
2	RW	0x0	pid2_done PID2 Channel Status 1 means done

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	W1C	0x0	pid1_done PID1 Channel Status 1 means done
0	W1C	0x0	pid0_done PID0 Channel Status 1 means done

**TSP\_PTIx\_PID\_STS1**

Address: Operational Base + offset (0x0118)

PTI PID Channel Status 1 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	W1C	0x0	pid63_done PID63 Channel Status 1 means done
30	W1C	0x0	pid62_done PID62 Channel Status 1 means done
29	W1C	0x0	pid61_done PID61 Channel Status 1 means done
28	W1C	0x0	pid60_done PID60 Channel Status 1 means done
27	W1C	0x0	pid59_done PID59 Channel Status 1 means done
26	W1C	0x0	pid58_done PID58 Channel Status 1 means done
25	W1C	0x0	pid57_done PID57 Channel Status 1 means done
24	W1C	0x0	pid56_done PID56 Channel Status 1 means done
23	W1C	0x0	pid55_done PID55 Channel Status 1 means done
22	W1C	0x0	pid54_done PID54 Channel Status 1 means done
21	W1C	0x0	pid53_done PID53 Channel Status 1 means done

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
20	W1C	0x0	pid52_done PID52 Channel Status 1 means done
19	W1C	0x0	pid51_done PID51 Channel Status 1 means done
18	W1C	0x0	pid50_done PID51 Channel Status 1 means done
17	W1C	0x0	pid49_done PID49 Channel Status 1 means done
16	W1C	0x0	pid48_done PID48 Channel Status 1 means done
15	W1C	0x0	pid47_done PID47 Channel Status 1 means done
14	W1C	0x0	pid46_done PID46 Channel Status 1 means done
13	W1C	0x0	pid45_done PID45 Channel Status 1 means done
12	W1C	0x0	pid44_done PID44 Channel Status 1 means done
11	W1C	0x0	pid43_done PID43 Channel Status 1 means done
10	W1C	0x0	pid42_done PID42 Channel Status 1 means done
9	W1C	0x0	pid41_done PID41 Channel Status 1 means done
8	W1C	0x0	pid40_done PID40 Channel Status 1 means done
7	W1C	0x0	pid39_done PID39 Channel Status 1 means done
6	W1C	0x0	pid38_done PID38 Channel Status 1 means done

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	W1C	0x0	pid37_done PID37 Channel Status 1 means done
4	W1C	0x0	pid36_done PID36 Channel Status 1 means done
3	RW	0x0	pid35_done PID35 Channel Status 1 means done
2	W1C	0x0	pid34_done PID34 Channel Status 1 means done
1	W1C	0x0	pid33_done PID33 Channel Status 1 means done
0	RW	0x0	pid32_done PID32 Channel Status 1 means done

**TSP\_PTIX\_PID\_STS2**

Address: Operational Base + offset (0x011c)

PTI PID Channel Status 2 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	pid31_error PID31 Error Interrupt Status 1 means error detected
30	W1C	0x0	pid30_error PID30 Error Interrupt Status 1 means error detected
29	W1C	0x0	pid29_error PID29 Error Interrupt Status 1 means error detected
28	W1C	0x0	pid28_error PID28 Error Interrupt Status 1 means error detected
27	W1C	0x0	pid27_error PID27 Error Interrupt Status 1 means error detected
26	W1C	0x0	pid26_error PID26 Error Interrupt Status 1 means error detected
25	W1C	0x0	pid25_error PID25 Error Interrupt Status 1 means error detected

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
24	W1C	0x0	pid24_error PID24 Error Interrupt Status 1 means error detected
23	W1C	0x0	pid23_error PID23 Error Interrupt Status 1 means error detected
22	W1C	0x0	pid22_error PID22 Error Interrupt Status 1 means error detected
21	W1C	0x0	pid21_error PID21 Error Interrupt Status 1 means error detected
20	W1C	0x0	pid20_error PID20 Error Interrupt Status 1 means error detected
19	W1C	0x0	pid19_error PID19 Error Interrupt Status 1 means error detected
18	W1C	0x0	pid18_error PID18 Error Interrupt Status 1 means error detected
17	W1C	0x0	pid17_error PID17 Error Interrupt Status 1 means error detected
16	W1C	0x0	pid16_error PID16 Error Interrupt Status 1 means error detected
15	W1C	0x0	pid15_error PID15 Error Interrupt Status 1 means error detected
14	W1C	0x0	pid14_error PID14 Error Interrupt Status 1 means error detected
13	W1C	0x0	pid13_error PID13 Error Interrupt Status 1 means error detected
12	W1C	0x0	pid12_error PID12 Error Interrupt Status 1 means error detected
11	W1C	0x0	pid11_error PID11 Error Interrupt Status 1 means error detected
10	W1C	0x0	pid10_error PID10 Error Interrupt Status 1 means error detected

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	W1C	0x0	pid9_error PID9 Error Interrupt Status 1 means error detected
8	W1C	0x0	pid8_error PID8 Error Interrupt Status 1 means error detected
7	W1C	0x0	pid7_error PID7 Error Interrupt Status 1 means error detected
6	W1C	0x0	pid6_error PID6 Error Interrupt Status 1 means error detected
5	W1C	0x0	pid5_error PID5 Error Interrupt Status 1 means error detected
4	W1C	0x0	pid4_error PID4 Error Interrupt Status 1 means error detected
3	W1C	0x0	pid3_error PID3 Error Interrupt Status 1 means error detected
2	W1C	0x0	pid2_error PID2 Error Interrupt Status 1 means error detected
1	W1C	0x0	pid1_error PID1 Error Interrupt Status 1 means error detected
0	W1C	0x0	pid0_error PID0 Error Interrupt Status 1 means error detected

**TSP\_PTIX\_PID\_STS3**

Address: Operational Base + offset (0x0120)

PTI PID Channel Status 3 Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	W1C	0x0	pid63_error PID63 Error Interrupt Status
30	W1C	0x0	pid62_error PID62 Error Interrupt Status
29	W1C	0x0	pid61_error PID61 Error Interrupt Status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
28	W1C	0x0	pid60_error PID60 Error Interrupt Status
27	W1C	0x0	pid59_error PID59 Error Interrupt Status
26	W1C	0x0	pid58_error PID58 Error Interrupt Status
25	W1C	0x0	pid57_error PID57 Error Interrupt Status
24	W1C	0x0	pid56_error PID56 Error Interrupt Status
23	W1C	0x0	pid55_error PID55 Error Interrupt Status
22	W1C	0x0	pid54_error PID54 Error Interrupt Status
21	W1C	0x0	pid53_error PID53 Error Interrupt Status
20	W1C	0x0	pid52_error PID52 Error Interrupt Status
19	W1C	0x0	pid51_error PID51 Error Interrupt Status
18	W1C	0x0	pid50_error PID50 Error Interrupt Status
17	W1C	0x0	pid49_error PID49 Error Interrupt Status
16	W1C	0x0	pid48_error PID48 Error Interrupt Status
15	W1C	0x0	pid47_error PID47 Error Interrupt Status
14	W1C	0x0	pid46_error PID46 Error Interrupt Status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13	W1C	0x0	pid45_error PID45 Error Interrupt Status
12	W1C	0x0	pid44_error PID44 Error Interrupt Status
11	W1C	0x0	pid43_error PID43 Error Interrupt Status
10	W1C	0x0	pid42_error PID42 Error Interrupt Status
9	W1C	0x0	pid41_error PID41 Error Interrupt Status
8	W1C	0x0	pid40_error PID40 Error Interrupt Status
7	W1C	0x0	pid39_error PID39 Error Interrupt Status
6	W1C	0x0	pid38_error PID38 Error Interrupt Status
5	W1C	0x0	pid37_error PID37 Error Interrupt Status
4	W1C	0x0	pid36_error PID36 Error Interrupt Status
3	W1C	0x0	pid35_error PID35 Error Interrupt Status
2	W1C	0x0	pid34_error PID34 Error Interrupt Status
1	W1C	0x0	pid33_error PID33 Error Interrupt Status
0	W1C	0x0	pid32_error PID32 Error Interrupt Status

**TSP\_PTIX\_PID\_INT\_ENAO**

Address: Operational Base + offset (0x0124)

## PID Interrupt Enable Register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	pid31_done_ena PID31 Done Enable 1:enabled 0:disabled
30	RW	0x0	pid30_done_ena PID30 Done Enable 1:enabled 0:disabled
29	RW	0x0	pid29_done_ena PID29 Done Enable 1:enabled 0:disabled
28	RW	0x0	pid28_done_ena PID28 Done Enable 1:enabled 0:disabled
27	RW	0x0	pid27_done_ena PID27 Done Enable 1:enabled 0:disabled
26	RW	0x0	pid26_done_ena PID26 Done Enable 1:enabled 0:disabled
25	RW	0x0	pid25_done_ena PID25 Done Enable 1:enabled 0:disabled
24	RW	0x0	pid24_done_ena PID24 Done Enable 1:enabled 0:disabled
23	RW	0x0	pid23_done_ena PID23 Done Enable 1:enabled 0:disabled
22	RW	0x0	pid22_done_ena PID22 Done Enable 1:enabled 0:disabled
21	RW	0x0	pid21_done_ena PID21 Done Enable 1:enabled 0:disabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
20	RW	0x0	pid20_done_ena PID20 Done Enable 1:enabled 0:disabled
19	RW	0x0	pid19_done_ena PID19 Done Enable 1:enabled 0:disabled
18	RW	0x0	pid18_done_ena PID18 Done Enable 1:enabled 0:disabled
17	RW	0x0	pid17_done_ena PID17 Done Enable
16	RW	0x0	pid16_done_ena PID16 Done Enable 1:enabled 0:disabled
15	RW	0x0	pid15_done_ena PID15 Done Enable 1:enabled 0:disabled
14	RW	0x0	pid14_done_ena PID14 Done Enable 1:enabled 0:disabled
13	RW	0x0	pid13_done_ena PID13 Done Enable 1:enabled 0:disabled
12	RW	0x0	pid12_done_ena PID12 Done Enable 1:enabled 0:disabled
11	RW	0x0	pid11_done_ena PID11 Done Enable 1:enabled 0:disabled
10	RW	0x0	pid10_done_ena PID10 Done Enable 1:enabled 0:disabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	pid9_done_ena PID9 Done Enable 1:enabled 0:disabled
8	RW	0x0	pid8_done_ena PID8 Done Enable 1:enabled 0:disabled
7	RW	0x0	pid7_done_ena PID7 Done Enable 1:enabled 0:disabled
6	RW	0x0	pid6_done_ena PID6 Done Enable 1:enabled 0:disabled
5	RW	0x0	pid5_done_ena PID5 Done Enable 1:enabled 0:disabled
4	RW	0x0	pid4_done_ena PID4 Done Enable 1:enabled 0:disabled
3	RW	0x0	pid3_done_ena PID3 Done Enable 1:enabled 0:disabled
2	RW	0x0	pid2_done_ena PID2 Done Enable 1:enabled 0:disabled
1	RW	0x0	pid1_done_ena PID1 Done Enable 1:enabled 0:disabled
0	RW	0x0	pid0_done_ena PID0 Done Enable 1:enabled 0:disabled

**TSP\_PTIX\_PID\_INT\_ENA1**

Address: Operational Base + offset (0x0128)

PID Interrupt Enable Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	pid63_done PID63 Done Enable 1:enabled 0:disabled
30	RW	0x0	pid62_done PID62 Done Enable 1:enabled 0:disabled
29	RW	0x0	pid61_done PID61 Done Enable 1:enabled 0:disabled
28	RW	0x0	pid60_done PID60 Done Enable 1:enabled 0:disabled
27	RW	0x0	pid59_done PID59 Done Enable 1:enabled 0:disabled
26	RW	0x0	pid58_done PID58 Done Enable 1:enabled 0:disabled
25	RW	0x0	pid57_done PID57 Done Enable 1:enabled 0:disabled
24	RW	0x0	pid56_done PID56 Done Enable 1:enabled 0:disabled
23	RW	0x0	pid55_done PID55 Done Enable 1:enabled 0:disabled
22	RW	0x0	pid54_done PID54 Done Enable 1:enabled 0:disabled
21	RW	0x0	pid53_done PID53 Done Enable 1:enabled 0:disabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
20	RW	0x0	pid52_done PID52 Done Enable 1:enabled 0:disabled
19	RW	0x0	pid51_done PID51 Done Enable 1:enabled 0:disabled
18	RW	0x0	pid50_done PID50 Done Enable 1:enabled 0:disabled
17	RW	0x0	pid49_done PID49 Done Enable 1:enabled 0:disabled
16	RW	0x0	pid48_done PID48 Done Enable 1:enabled 0:disabled
15	RW	0x0	pid47_done PID47 Done Enable 1:enabled 0:disabled
14	RW	0x0	pid46_done PID46 Done Enable 1:enabled 0:disabled
13	RW	0x0	pid45_done PID45 Done Enable 1:enabled 0:disabled
12	RW	0x0	pid44_done PID44 Done Enable 1:enabled 0:disabled
11	RW	0x0	pid43_done PID43 Done Enable 1:enabled 0:disabled
10	RW	0x0	pid42_done PID42 Done Enable 1:enabled 0:disabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	pid41_done PID41 Done Enable 1:enabled 0:disabled
8	RW	0x0	pid40_done PID40 Done Enable 1:enabled 0:disabled
7	RW	0x0	pid39_done PID39 Done Enable 1:enabled 0:disabled
6	RW	0x0	pid38_done PID38 Done Enable 1:enabled 0:disabled
5	RW	0x0	pid37_done PID37 Done Enable 1:enabled 0:disabled
4	RW	0x0	pid36_done PID36 Done Enable 1:enabled 0:disabled
3	RW	0x0	pid35_done PID35 Done Enable 1:enabled 0:disabled
2	RW	0x0	pid34_done PID34 Done Enable 1:enabled 0:disabled
1	RW	0x0	pid33_done PID33 Done Enable 1:enabled 0:disabled
0	RW	0x0	pid32_done PID32 Done Enable 1:enabled 0:disabled

**TSP\_PTIX\_PID\_INT\_ENA2**

Address: Operational Base + offset (0x012c)

PID Interrupt Enable Register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	pid31_error PID31 Error Interrupt Enable 1:enabled 0:disabled
30	RW	0x0	pid30_error PID30 Error Interrupt Enable 1:enabled 0:disabled
29	RW	0x0	pid29_error PID29 Error Interrupt Enable 1:enabled 0:disabled
28	RW	0x0	pid28_error PID28 Error Interrupt Enable 1:enabled 0:disabled
27	RW	0x0	pid27_error PID27 Error Interrupt Enable 1:enabled 0:disabled
26	RW	0x0	pid26_error PID26 Error Interrupt Enable 1:enabled 0:disabled
25	RW	0x0	pid25_error PID25 Error Interrupt Enable 1:enabled 0:disabled
24	RW	0x0	pid24_error PID24 Error Interrupt Enable 1:enabled 0:disabled
23	RW	0x0	pid23_error PID23 Error Interrupt Enable 1:enabled 0:disabled
22	RW	0x0	pid22_error PID22 Error Interrupt Enable 1:enabled 0:disabled
21	RW	0x0	pid21_error PID21 Error Interrupt Enable 1:enabled 0:disabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
20	RW	0x0	pid20_error PID20 Error Interrupt Enable 1:enabled 0:disabled
19	RW	0x0	pid19_error PID19 Error Interrupt Enable 1:enabled 0:disabled
18	RW	0x0	pid18_error PID18 Error Interrupt Enable 1:enabled 0:disabled
17	RW	0x0	pid17_error PID17 Error Interrupt Enable 1:enabled 0:disabled
16	RW	0x0	pid16_error PID16 Error Interrupt Enable 1:enabled 0:disabled
15	RW	0x0	pid15_error PID15 Error Interrupt Enable 1:enabled 0:disabled
14	RW	0x0	pid14_error PID14 Error Interrupt Enable 1:enabled 0:disabled
13	RW	0x0	pid13_error PID13 Error Interrupt Enable 1:enabled 0:disabled
12	RW	0x0	pid12_error PID12 Error Interrupt Enable 1:enabled 0:disabled
11	RW	0x0	pid11_error PID11 Error Interrupt Enable 1:enabled 0:disabled
10	RW	0x0	pid10_error PID10 Error Interrupt Enable 1:enabled 0:disabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	pid9_error PID9 Error Interrupt Enable 1:enabled 0:disabled
8	RW	0x0	pid8_error PID8 Error Interrupt Enable 1:enabled 0:disabled
7	RW	0x0	pid7_error PID7 Error Interrupt Enable 1:enabled 0:disabled
6	RW	0x0	pid6_error PID6 Error Interrupt Enable 1:enabled 0:disabled
5	RW	0x0	pid5_error PID5 Error Interrupt Enable 1:enabled 0:disabled
4	RW	0x0	pid4_error PID4 Error Interrupt Enable 1:enabled 0:disabled
3	RW	0x0	pid3_error PID3 Error Interrupt Enable 1:enabled 0:disabled
2	RW	0x0	pid2_error PID2 Error Interrupt Enable 1:enabled 0:disabled
1	RW	0x0	pid1_error PID1 Error Interrupt Enable 1:enabled 0:disabled
0	RW	0x0	pid0_error PID0 Error Interrupt Enable 1:enabled 0:disabled

**TSP\_PTIX\_PID\_INT\_ENA3**

Address: Operational Base + offset (0x0130)

PID Interrupt Enable Register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	pid63_error PID63 Error Interrupt Enable 1:enabled 0:disabled
30	RW	0x0	pid62_error PID62 Error Interrupt Enable 1:enabled 0:disabled
29	RW	0x0	pid61_error PID61 Error Interrupt Enable 1:enabled 0:disabled
28	RW	0x0	pid60_error PID60 Error Interrupt Enable 1:enabled 0:disabled
27	RW	0x0	pid59_error PID59 Error Interrupt Enable 1:enabled 0:disabled
26	RW	0x0	pid58_error PID58 Error Interrupt Enable 1:enabled 0:disabled
25	RW	0x0	pid57_error PID57 Error Interrupt Enable 1:enabled 0:disabled
24	RW	0x0	pid56_error PID56 Error Interrupt Enable 1:enabled 0:disabled
23	RW	0x0	pid55_error PID55 Error Interrupt Enable 1:enabled 0:disabled
22	RW	0x0	pid54_error PID54 Error Interrupt Enable 1:enabled 0:disabled
21	RW	0x0	pid53_error PID53 Error Interrupt Enable 1:enabled 0:disabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
20	RW	0x0	pid52_error PID52 Error Interrupt Enable 1:enabled 0:disabled
19	RW	0x0	pid51_error PID51 Error Interrupt Enable 1:enabled 0:disabled
18	RW	0x0	pid50_error PID50 Error Interrupt Enable 1:enabled 0:disabled
17	RW	0x0	pid49_error PID49 Error Interrupt Enable 1:enabled 0:disabled
16	RW	0x0	pid48_error PID48 Error Interrupt Enable 1:enabled 0:disabled
15	RW	0x0	pid47_error PID47 Error Interrupt Enable 1:enabled 0:disabled
14	RW	0x0	pid46_error PID46 Error Interrupt Enable 1:enabled 0:disabled
13	RW	0x0	pid45_error PID45 Error Interrupt Enable 1:enabled 0:disabled
12	RW	0x0	pid44_error PID44 Error Interrupt Enable 1:enabled 0:disabled
11	RW	0x0	pid43_error PID43 Error Interrupt Enable 1:enabled 0:disabled
10	RW	0x0	pid42_error PID42 Error Interrupt Enable 1:enabled 0:disabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	pid41_error PID41 Error Interrupt Enable 1:enabled 0:disabled
8	RW	0x0	pid40_error PID40 Error Interrupt Enable 1:enabled 0:disabled
7	RW	0x0	pid39_error PID39 Error Interrupt Enable 1:enabled 0:disabled
6	RW	0x0	pid38_error PID38 Error Interrupt Enable 1:enabled 0:disabled
5	RW	0x0	pid37_error PID37 Error Interrupt Enable 1:enabled 0:disabled
4	RW	0x0	pid36_error PID36 Error Interrupt Enable 1:enabled 0:disabled
3	RW	0x0	pid35_error PID35 Error Interrupt Enable 1:enabled 0:disabled
2	RW	0x0	pid34_error PID34 Error Interrupt Enable 1:enabled 0:disabled
1	RW	0x0	pid33_error PID33 Error Interrupt Enable 1:enabled 0:disabled
0	RW	0x0	pid32_error PID32 Error Interrupt Enable 1:enabled 0:disabled

**TSP\_PTIx\_PCR\_INT\_STS**

Address: Operational Base + offset (0x0134)

PTI PCR Interrupt Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7	W1C	0x0	pcr7_done PCR7 Status 1: done; 0: not done;
6	W1C	0x0	pcr6_done PCR6 Status 1: done; 0: not done;
5	W1C	0x0	pcr5_done PCR5 Status 1: done; 0: not done;
4	W1C	0x0	pcr4_done PCR4 Status 1: done; 0: not done;
3	W1C	0x0	pcr3_done PCR3 Status 1: done; 0: not done;
2	W1C	0x0	pcr2_done PCR2 Status 1: done; 0: not done;
1	W1C	0x0	pcr1_done PCR1 Status 1: done; 0: not done;
0	W1C	0x0	pcr0_done PCR0 Status 1: done; 0: not done;

**TSP\_PTIx\_PCR\_INT\_ENA**

Address: Operational Base + offset (0x0138)

PTI PCR Interrupt Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7	RW	0x0	pcr7_done_ena pcr7 done interrupt enable 1: enabled; 0: disabled;

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RW	0x0	pcr6_done_ena pcr6 done interrupt enable 1: enabled; 0: disabled;
5	RW	0x0	pcr5_done_ena pcr5 done interrupt enable 1: enabled; 0: disabled;
4	RW	0x0	pcr4_done_ena pcr4 done interrupt enable 1: enabled; 0: disabled;
3	RW	0x0	pcr3_done_ena pcr3 done interrupt enable 1: enabled; 0: disabled;
2	RW	0x0	pcr2_done_ena pcr2 done interrupt enable 1: enabled; 0: disabled;
1	RW	0x0	pcr1_done_ena pcr1 done interrupt enable 1: enabled; 0: disabled;
0	RW	0x0	pcr0_done_ena pcr0 done interrupt enable 1: enabled; 0: disabled;

**TSP\_PTIx\_PCRn\_CTRL**

Address: Operational Base + offset (0x013c)

PID PCR Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:14	RO	0x0	reserved
13:1	RW	0x0000	pid PCR Extraction PID number This 13-bit field sets the PID number that needs PCR extraction.
0	RW	0x0	on PCR Extraction Switch 1'b1: PCR extraction switched on ; 1'b0: PCR extraction switched off ;

**TSP\_PTIx\_PCRn\_H**

Address: Operational Base + offset (0x015c)

High Order PCR value

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RO	0x0	pcr PCR[32] pcr[32]

### TSP\_PTIX\_PCRn\_L

Address: Operational Base + offset (0x0160)

Low Order PCR value

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	pcr pcr[31:0] pcr[31:0]

### TSP\_PTIX\_DMA\_STS

Address: Operational Base + offset (0x019c)

LLP DMA Interrupt Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	W1C	0x0	llp_error LLP DMA Error Status 1: error response during DMA transaction; 0: no error response during DMA transaction;
0	W1C	0x0	llp_done LLP DMA Done Status 1: DMA transaction completed; 0: DMA transaction not completed;

### TSP\_PTIX\_DMA\_ENA

Address: Operational Base + offset (0x01a0)

DMA Interrupt Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	llp_error_ena LLP DMA Error Interrupt Enable 1: enabled 0: disabled
0	RW	0x0	llp_done_ena LLP DMA Done Interrupt Enable 1: enabled 0: disabled

### TSP\_PTIX\_DATA\_FLAG0

Address: Operational Base + offset (0x01a4)

PTI\_PID\_WRITE Flag 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	data_write_flag_0 From PID0 TO PID31

**TSP\_PTIx\_DATA\_FLAG1**

Address: Operational Base + offset (0x01a8)

PTI\_PID\_WRITE Flag 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	data_write_flag_1 From PID32 TO PID63

**TSP\_PTIx\_LIST\_FLAG**

Address: Operational Base + offset (0x01ac)

PTIx\_LIST\_WRITE Flag

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	list_write_flag From PID0 TO PID15

**TSP\_PTIx\_DST\_STS0**

Address: Operational Base + offset (0x01b0)

PTI Destination Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	W1C	0x00000000	demux_dma_status_0 From 0 to 31 channel

**TSP\_PTIx\_DST\_STS1**

Address: Operational Base + offset (0x01b4)

PTI Destination Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	W1C	0x00000000	demux_dma_status_0 From 32 to 63 channel

**TSP\_PTIx\_DST\_ENA0**

Address: Operational Base + offset (0x01b8)

PTI Destination Interrupt Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	demux_dma_enable_0 From 0 to 31 channel

**TSP\_PTIx\_DST\_ENA1**

Address: Operational Base + offset (0x01bc)

PTI Destination Interrupt Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	demux_dma_enable_1 From 32 to 63 channel

**TSP\_PTIx\_ECWn\_H**

Address: Operational Base + offset (0x0200)

The Even Control Word High Order

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	ecw_h The Even Control Word High Order ECW[63:32]

**TSP\_PTIx\_ECWn\_L**

Address: Operational Base + offset (0x0204)

The Even Control Word Low Order

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	ecw_l The Even Control Word Low Order ECW[31:0]

**TSP\_PTIx\_OCWn\_H**

Address: Operational Base + offset (0x0208)

The Odd Control Word High Order

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	ocw_h The Odd Control Word High order OCW[63:32]

**TSP\_PTIx\_OCWn\_L**

Address: Operational Base + offset (0x020c)

The Odd Control Word Low Order

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	ocw_l The Odd Control Word Low Order OCW[31:0]

**TSP\_PTIx\_PIDn\_CTRL**

Address: Operational Base + offset (0x0300)

PID Channel Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19:16	RW	0x0	cw_num Control Word Order Number This fields indicates the corresponding order number of control word to be used to descramble TS packets.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:3	RW	0x0000	pid PID number This 13-bit sets the desired PID number to be processed by PTI channel.
2	RW	0x0	csa_on Descrambling Switch 1'b1: Descrambling function turned on; 1'b0: Descrambling function turned off;
1	RWSC	0x0	clear PID Channel Clear Write 1 to clear PID channel. This bit will be set to 0 if the channel is clear.
0	RWSC	0x0	en PID Channel Enable Write 1 to enable channel. Write 0 to this bit will not take any effect. This bit will be 0 when channel is cleared.

**TSP\_PTIX\_PIDn\_BASE**

Address: Operational Base + offset (0x0400)

PTI Data Memory Buffer Base Address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	address PTI Data Memory Buffer Base Address PTI Data Memory Buffer Base Address

**TSP\_PTIX\_PIDn\_TOP**

Address: Operational Base + offset (0x0404)

PTI Data Memory Buffer Top Address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	address PTI Data Memory Buffer Top Address PTI Data Memory Buffer Top Address

**TSP\_PTIX\_PIDn\_WRITE**

Address: Operational Base + offset (0x0408)

PTI Data Memory Buffer Hardware Writing Address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	address PTI Data Memory Buffer Hardware Writing Address PTI Data Memory Buffer Hardware Writing Address

**TSP\_PTIx\_PIDn\_READ**

Address: Operational Base + offset (0x040c)

PTI Data Memory Buffer Software Reading Address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	address PTI Data Memory Buffer Software Reading Address PTI Data Memory Buffer Software Reading Address

**TSP\_PTIx\_LISTn\_BASE**

Address: Operational Base + offset (0x0800)

PTI List Memory Buffer Base Address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	address PTI Data Memory Buffer Software Reading Address PTI Data Memory Buffer Software Reading Address

**TSP\_PTIx\_LISTn\_TOP**

Address: Operational Base + offset (0x0804)

PTI List Memory Buffer Top Address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	address PTI List Memory Buffer Top Address PTI List Memory Buffer Top Address

**TSP\_PTIx\_LISTn\_WRITE**

Address: Operational Base + offset (0x0808)

PTI List Memory Buffer Hardware Writing Address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	address PTI List Memory Buffer Hardware Writing Address PTI List Memory Buffer Hardware Writing Address

**TSP\_PTIx\_LISTn\_READ**

Address: Operational Base + offset (0x080c)

PTI List Memory Buffer Software Reading Address

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	address PTI List Memory Buffer Software Reading Address PTI List Memory Buffer Software Reading Address

**TSP\_PTIX\_PIDn\_CFG**

Address: Operational Base + offset (0x0900)

PID Demux Configure Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0000	filter_en Filter Byte Enable The proper position of filter byte Enable. For Section filter. the 1st,4th,5th,..18th byte of section header are used to be filtered; For PES filter, the 4th,7th,8th...21th byte of pes header are used to be filtered.
15:12	RO	0x0	reserved
11	RW	0x0	scd_en Start Code Detection Switch Start code detection 1: enabled; 0: disabled; This bit is only valid when n < 16.
10	RW	0x0	cni_on Current Next Indicator Abort when current_next_indicator == 1'b1, 1'b1: abort ; 1'b0: do nothing ;
9:8	RW	0x0	filt_mode Section Filter Mode Filter Mode when the filter mode is configured as section filter. 2'b00: stop per unit; 2'b01: full stop; 2'b10: recycle, update when version number change 2'b11: reserved
7:6	RW	0x0	video_type Video filtering Type 2'b00: MPEG2 2'b01: H264 2'b10: VC-1 2'b11: Reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:4	RW	0x0	filt_type Filter Type 2'b00: section filtering; 2'b01: pes filtering; 2'b10: es filtering; 2'b11: ts filtering; if n>=16, it is reserved as only section filtering, other values are invalid.
3	RW	0x1	cc_abort Continue Counter Error Abort when continuity counter error happens: 1: abort; 0: do nothing;
2	RW	0x0	tei_abort Ts_error_indicator Abort when ts_error_indicator == 1: 1'b1: abort ; 1'b0: do nothing;
1	RW	0x0	crc_abort CRC Error Abort This bit is valid only when crc_on == 1'b1. When crc error happens, 1'b1: abort ; 1'b0: do nothing.
0	RW	0x0	crc_on CRC Check 1'b1: CRC check function turned on 1'b0: CRC check function turned off

**TSP\_PTIX\_PIDn\_FILT\_0**

Address: Operational Base + offset (0x0904)

Fliter Word 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	filt_byte_3 Fliter Byte 2 This byte refers to 6th byte of section header or 9th byte of pes header
23:16	RW	0x00	filt_byte_2 Fliter Byte 2 This byte refers to 5th byte of section header or 8th byte of pes header
15:8	RW	0x00	filt_byte_1 Fliter Byte 1 This byte refers to 4th byte of section header or 7th byte of pes header

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	RW	0x00	filt_byte_0 Fliter Byte 0 This byte refers to 1st byte of section header or 4th byte of pes header

**TSP\_PTIx\_PIDn\_FILT\_1**

Address: Operational Base + offset (0x0908)

Fliter Word 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	filt_byte_3 Fliter Byte 2 This byte refers to 10th byte of section header or 13rd byte of pes header
23:16	RW	0x00	filt_byte_2 Fliter Byte 2 This byte refers to 9th byte of section header or 12nd byte of pes header
15:8	RW	0x00	filt_byte_1 Fliter Byte 1 This byte refers to 8th byte of section header or 11st byte of pes header
7:0	RW	0x00	filt_byte_0 Fliter Byte 0 This byte refers to 7th byte of section header or 10th byte of pes header

**TSP\_PTIx\_PIDn\_FILT\_2**

Address: Operational Base + offset (0x090c)

Fliter Word 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	filt_byte_3 Fliter Byte 2 This byte refers to 14th byte of section header or 17th byte of pes header
23:16	RW	0x00	filt_byte_2 Fliter Byte 2 This byte refers to 13rd byte of section header or 16th byte of pes header
15:8	RW	0x00	filt_byte_1 Fliter Byte 1 This byte refers to 12nd byte of section header or 15th byte of pes header

Bit	Attr	Reset Value	Description
7:0	RW	0x00	filt_byte_0 Fliter Byte 0 This byte refers to 11st byte of section header or 14th byte of pes header

**TSP\_PTIx\_PIDn\_FILT\_3**

Address: Operational Base + offset (0x0910)

Fliter Word 3

Bit	Attr	Reset Value	Description
31:24	RW	0x00	filt_byte_3 Fliter Byte 2 This byte refers to 18th byte of section header or 21st byte of pes header
23:16	RW	0x00	filt_byte_2 Fliter Byte 2 This byte refers to 17th byte of section header or 20th byte of pes header
15:8	RW	0x00	filt_byte_1 Fliter Byte 1 This byte refers to 16th byte of section header or 19th byte of pes header
7:0	RW	0x00	filt_byte_0 Fliter Byte 0 This byte refers to 15th byte of section header or 18th byte of pes header

**39.5 Interface Description**

Table 39-1 TSP Interface Description

Module Pin	IO	Pad Name	IOMUX Setting
ts_data0	I/O	IO_UART1BBsin_TS0data0_BBgpio5b0	GPIO5B_IOMUX[1:0]= 2'b10
ts_data1	I/O	IO_UART1BBsout_TS0data1_BBgpio5b1	GPIO5B_IOMUX[3:2]= 2'b10
ts_data2	I/O	IO_UART1BBctsn_TS0data2_BBgpio5b2	GPIO5B_IOMUX[5:4]= 2'b10
ts_data3	I/O	IO_UART1BBrtsn_TS0data3_BBgpio5b3	GPIO5B_IOMUX[7:6]= 2'b10
ts_data4	I/O	IO_SPI0clk_TS0data4_UART4EXPcts_BBgpio5b4	GPIO5B_IOMUX[9:8]= 2'b10
ts_data5	I/O	IO_SPI0csn0_TS0data5_UART4EXPrtsn_BBgpio5b5	GPIO5C_IOMUX[11:10]= 2'b10
ts_data6	I/O	IO_SPI0txd_TS0data6_UART4EXPouts_BBgpio5b6	GPIO5B_IOMUX[13:12]= 2'b10
ts_data7	I/O	IO_SPI0rxr_TS0data7_UART4EXPsin_BBgpio5b7	GPIO5B_IOMUX[15:14]= 2'b10
ts_valid	I/O	IO_TS0valid_BBgpio5c1	GPIO5C_IOMUX[2]= 1'b1
ts_sync	I/O	IO_SPI0csn1_TS0sync_BBgpio5c0	GPIO5C_IOMUX[1:0]= 2'b10
ts_err	I/O	IO_TS0err_BBgpio5c3	GPIO5C_IOMUX[6]= 1'b1
ts_clk	I/O	IO_TS0clk_BBgpio5c2	GPIO5C_IOMUX[4]= 1'b1
hsadc_data0	I	IO_CIFdata2_HOSTdin0_HSADCdata0_DVPgpio2a0	GPIO2A_IOMUX[1:0]= 2'b11
hsadc_data1	I	IO_CIFdata3_HOSTdin1_HSADCdata1_DVPgpio2a1	GPIO2A_IOMUX[3:2]= 2'b11
hsadc_data2	I	IO_CIFdata4_HOSTdin2_HSADCdata2_DVPgpio2a2	GPIO2A_IOMUX[5:4]= 2'b11
hsadc_data3	I	IO_CIFdata5_HOSTdin3_HSADCdata3_DVPgpio2a3	GPIO2A_IOMUX[7:6]= 2'b11
hsadc_data4	I	IO_CIFdata6_HOSTckinp_HSADCdata4_DVPgpio2a4	GPIO2A_IOMUX[9:8]= 2'b11
hsadc_data5	I	IO_CIFdata7_HOSTckinn_HSADCdata5_DVPgpio2a5	GPIO2A_IOMUX[11:10]= 2'b11
hsadc_data6	I	IO_CIFdata8_HOSTdin4_HSADCdata6_DVPgpio2a6	GPIO2A_IOMUX[13:12]= 2'b11
hsadc_data7	I	IO_CIFdata9_HOSTdin5_HSADCdata7_DVPgpio2a7	GPIO2A_IOMUX[15:14]= 2'b11
hsadc_valid	I	IO_CIFhref_HOSTdin7_HSADCTSvalid_DVPgpio2b1	GPIO2B_IOMUX[3:2]= 2'b11

hsadc_sync	I	IO_CIFVsync_HOSTdin6_HSADCCTSsync_DVPgpio2b0	GPIO2B_IOMUX[1:0]= 2'b11
hsadc_err	I	IO_CIFClkout_HOSTwkreq_HSADCCTSfail_DVPgpio2b3	GPIO2B_IOMUX[7:6]= 2'b01
gps_clk	I	IO_CIFClkin_HOSTwkack_GPSclk_HSADCclkout_DVPgpio2b2	GPIO2B_IOMUX[5:4]= 2'b11
gpst1_clk	I	IO_UART3GPScsn_GPSrfclk_GPST1clk_GPIO30gpio7b1	GPIO7B_IOMUX[3:2]= 2'b11

Notes: I=input, O=output, I/O=input/output, bidirectional

## 39.6 Application Notes

### 39.6.1 Overall Operation Sequence

- Enable desired modules to work by writing correspond bit with '1' in TSP\_GCFG. Note: it is important to do this step at first, otherwise writing the corresponding registers will not take effect.
- Set up TS configuration by writing corresponding registers.
- Wait for the interrupts to pick up the desired TS packets following the rules detailed in the following section.

Note: PTI1 addr = PTI0 addr + 0x1000;

### 39.6.2 TS Source

TS source can be chosen by writing the bit 9 of TSP\_PTIx\_CTRL(x=0,1), '1' for demodulator, '0' for local memory.

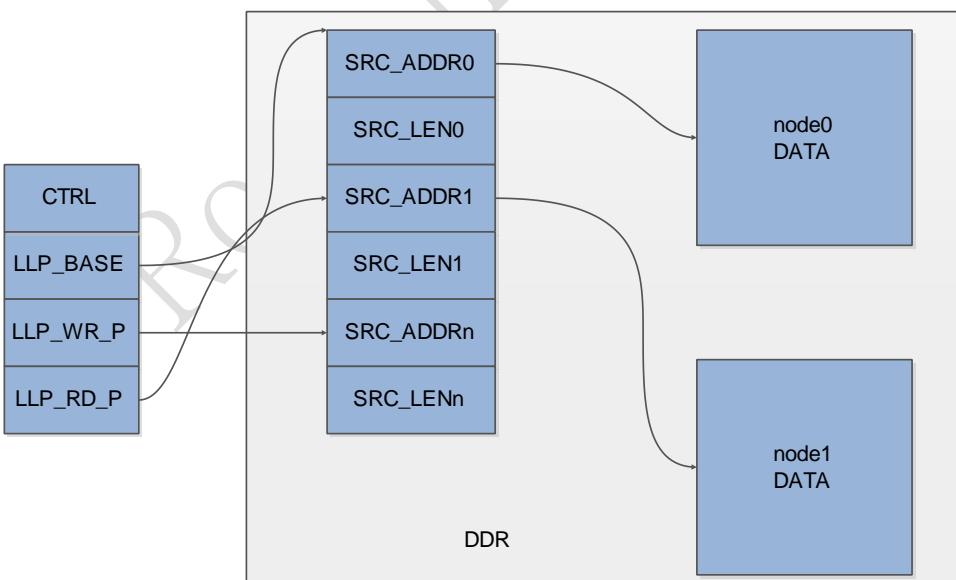
#### 1.TS\_IN Interface

Writing bit 10 of TSP\_PTIx\_CTRL to choose bit ordering, and writing bit [12:11] to choose input TS mode.

TS\_IN interface supports 4 input TS stream mode: sync/valid serial mode, sync/valid parallel mode, sync/burst parallel mode, nosync/valid parallel mode.

#### 2.Local Memory

PTI also can process the TS data read from local memory by using LLP DMA mode.



- (1) Write PTIx\_LL\_P BASE with the list base address;
- (2) Starting from the list base address, write the list nodes. One list node comprised of two words. The first word describes the TS data base address, the second one

describes the length of TS data in unit of word.

- (3) Write the PTIx\_LLW\_WRITE with the number of words that you have written in list memory. Note it is not the number of LLP nodes, so that the number you are writing should be an even one.
- (4) Write PTIx\_LLW\_CFG with the configuration you want. Write the bit 0 with 1 to start LLP DMA. If all the list nodes are written, don't forget to write 1 to bit 3 to tell DMAC that the configuration is finished.

*Note:*

- The MSB(bit7) of the 8-bit pointer in the PTIx\_LLW\_Write and PTIx\_LLW\_Read is used as the flag bit, and remaining 7 bits are used for addressing. Therefore the the pointer is referred to 7-bit space, not 8-bit space, and remember write the pointer with the correct flag bit. For example, if you have configured 63 LLP nodes and then you have to write the 64<sup>th</sup> LLP node starting from the list base address,
- PTIx\_LLW\_READ informs that how many words has been processed by LLP DMA. An interrupt may be generated when number of the processed words has reach to the threshold set in the PTIx\_LLW\_CFG.
- If you write the PTIx\_LLW\_Write several times in a complete DMA transaction, it is important to notice the flag bit of PTIx\_LLW\_Write, and never make the writing pointer catch up with the reading pointer.

### 39.6.3 TS Synchronous Operation

Synchronous mode and Bypass mode can be switched by writing bit 15 of TSP\_PTIx\_CTRL.

In the synchronous mode, 188/192/204 byte TS packets are supported and self-adjusted. Set up locked times in TSP\_PTIx\_CTRL to inform the successive times of TS packet header detection needs to lock the header of TS packets when in the unlocked mode, and set up unlocked times to informs the successive times of TS packet header error needs to re-lock header of TS packets in the locked mode. It is recommended to use 2-3 as the locked times to quickly and correctly locked the header, and 2-3 as unlocked times to avoid unnecessarily entering into unlocked searching mode.

In the bypass mode, the input TS data will not be re-synchronized and directly fed into the PTI channel.

### 39.6.4 Descrambling Operation

Descrambler can achieve PES or TS level descrambling which conforms to the CSA v2.0.

- Enable the channel you want by writing 1 to bit 0 of TSP\_PTIx\_PIDn\_CTRL (x=0~1, n=0~64);
- Set the desired PID number
- Turn on descrambling function by setting 1 to bit 2. If the corresponding CW is available or TS is required to be left undescrambled, CSA\_ON bit is set to 0;
- Choose corresponding Control Word by setting bit[19:16], and 16 set Control Word are available to be chosen. Don't forget Control Word should be prepared before the descrambling function is enabled.

*Note: If the enabled channel is needed to be disabled, write the CLEAR bit to disabled the channel rather than write '0' to EN bit.*

### 39.6.5 Demux Operation

Refer to TSP\_PTIX\_PIDn\_CFG for Demux operation. The software users should be familiar with the demux knowledge.

Users should create a separate memory buffer to receive the processed data for each desired PID channel, and write the base and top address information of the memory buffer into TSP\_PTIX\_PIDn\_BASE and TSP\_PTIX\_PIDn respectively. Also initial writing address and reading address, normally the same as base address, are also needed to be written into TSP\_PTIX\_PIDn\_WRITE and TSP\_PTIX\_PIDn\_READ respectively. For ES/PES filter, another separate memory needs to be created to store list data, which is used to assist obtaining PES/ES data. List base address, top address, initial writing address and reading address are also needed to write into corresponding registers.

*Note:*

1. For channel whose PID channel number larger than 15, the channels can only be used section filter. For others, there is no such limit. They can be configured as section filter, pes filter, es filter or ts filter.
2. Data memory address boundary should be aligned with word-size, and list memory address boundary should be aligned with word size. If the memory buffer is not larger to store processed data so that writing address reaches the top address, TSP will return to the base address to write data. So fetch the data in time, don't make the writing address catches up with reading address. The list memory buffer has the same issue.

#### 1. Demux data obtain

- **TS filter**

To obtain TS data and section data, when an desired PID done interrupt is generated, read TSP\_PTIX\_PIDn\_READ firstly to know the address that last reading stops, and then read TSP\_PTIX\_PIDn\_WRITE to know the address that hardware has reached. For ts data, start from the TSP\_PTIX\_PIDn\_READ address to get the TS packet data, and stop at the address you want. However, the ending address should not catch up with writing address. It is recommended to obtain the TS data in the unit of TS packet which is 47-word size. At last, don't forget to write the ending address into TSP\_PTIX\_PIDn\_READ to leave a hint where current reading stops.

#### B. Section filter

Section filter can run three mode to meet different needs: stop-per-unit; full stop; recycle , update when version number change. The PID done interrupt will be generated after each part of a complete section is processed in the first mode, and the PID done will be generated only after the whole section is completed in the last two modes. In the frist two mode, the PID channel will be disabled after the whole section is completed. In the recycle mode, the channel will remain active and start a new section processing when the version number changes. Section filter also supports 16-byte filtering function, which can assign 1<sup>st</sup> , 4<sup>th</sup> to 18<sup>th</sup> byte to be filtered.

The process to obtain section data is similar to the process for TS data. After a PID done interrupt done is generated, refer to the corresponding PID error status register to check if the section data is correct. Read the frist word of the section start address to know the total length of the section according to the format of section data.

$$\begin{aligned} \text{Section Length} &= \{\text{First Word}[11:8], \text{First Word}[23:16]\}; \\ \text{Total Length} &= \text{Section Length}; \end{aligned}$$

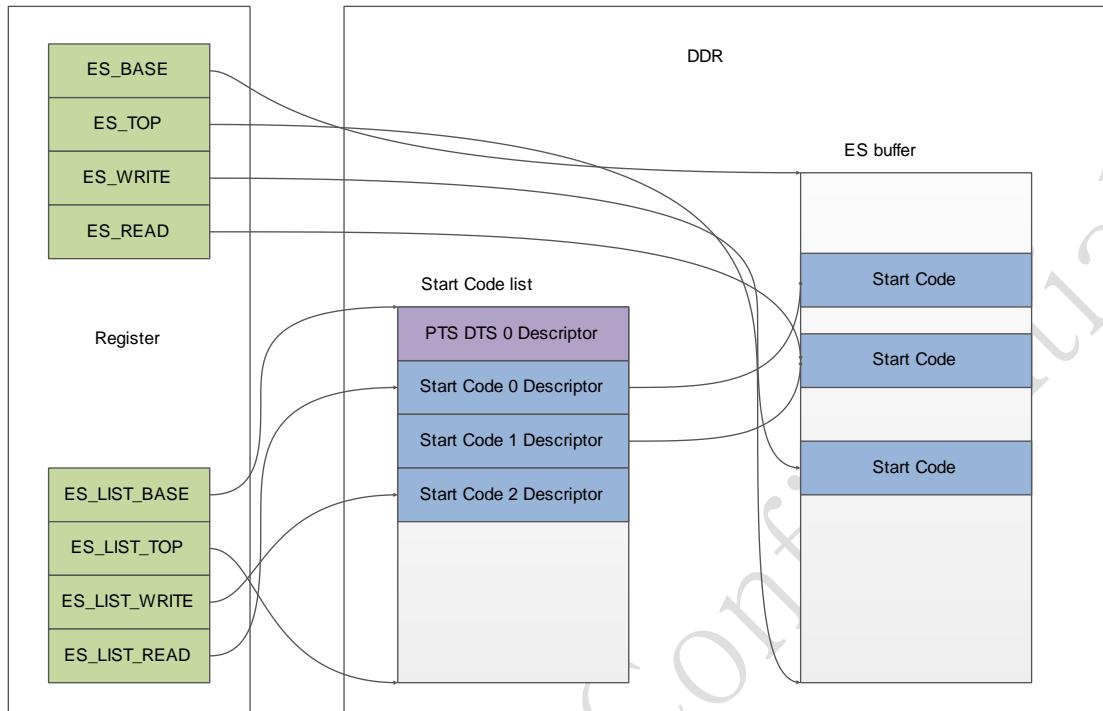
Then start to fetch section data according to the total length. Again don't forget to write the stopped address.

## C. PES/ES filter

PES filter supports 16-byte filtering function, which can assign 4<sup>th</sup>, 7<sup>th</sup> to 21<sup>st</sup> byte to be filtered.

ES filter supports start code detection, including MPEG2 start code 0x000001b3, 0x00000100, VC-1 start code 0x0000010d, 0x000010f, H264 start code 0x00001.

To obtain the pes/es data, the assistant of list descriptor is needed.



List memory buffer contains descriptors which contains information to obtain es/pes data which are stored in data memory buffer.

The descriptor stored in list memory buffer can be separated into two groups: PTS\_DTS Descriptor and Start Code Descriptor. The descriptor is composed by 4 word content, word\_0, word\_1, word\_2 and word\_3. The word\_x (x means the sequence number in a descriptor, and they are stored in the memory in sequence order). The format of the 4 words are listed as follows:

### (1) start code descriptor

Word\_0:

Word\_0[29:28] indicates the attributes of the bytes of the pointed word. 2'b00 means the whole word belongs to the new ES/PES packet; 2'b01 means that word[7:0] belongs to the previous packet, and the remaining bytes belong to the new packet; 2'b10 means means that word[15:0] belongs to the previous packet, and the remaining bytes belong to the new packet; 2'b11 means 'b10 means means that word[23:0] belongs to the previous packet, and the remaining bytes belong to the new packet. This pointed word is the word where start code starts, word\_2 describes the location of start code.

Word\_0[27:24] is equal to 0x0 in the start code descriptor. Users can used to tell two kinds of descriptor.

If the video type is H.264, word\_0[23:8] means first\_mb\_in slice, and word\_0 means nal\_nuit\_type.

Word\_1:

the start code of stream.

Word\_2:

DDR offset address in the DDR of the word where the start code is located.

Word\_3:

0x0

## **(2) PTS\_DTS Descriptor**

Word\_0:

Word\_0[29:28]: the same as start code descriptor  
Word\_0[27:24]: 0x1 in PTS\_DTS descriptor.  
Word\_0[3] : PTS[32];  
Word\_0[2] : DTS[32];  
Word\_0[1:0] : pts\_dts\_flag;

Word\_1:

DDR offset address of the word that valid data starts.

Word\_2:

PTS[31:0]

Word\_3

DTS[31:0]

To obtain PES data or ES data when start code detection is disabled, use PTS\_DTS descriptor.  
To obtain ES data when start code detection is enabled, use start code descriptor.

When a PID done interrupt is generated, make sure there is no corresponding PID error generated. Read the TSP\_PTIx\_LISTn\_READ to know the list reading address in the last time. Start from here, read the 4-word descriptor one by one to know the offset of the packets. Refer to the offset in the DDR where in the data memory buffer to obtain data. Finally write TSP\_PTIx\_LISTn\_READ and TSP\_PTIx\_PIDn\_READ with corresponding reading address.

## **39.6.6 TS Out Interface**

All the configuration is done by writing TSP\_TSOUT\_CTRL. Before programming this register, make sure that you have enabled the TS OUT interface. If you want to disable TS out interface, write '0' to the START bit(bit 0) of TSP\_TSOUT\_CTRL, and then disable it in the TSP\_GFCG.

Each PTI channel can provide TS out interface with PID-filtering TS Packets or non-PID-filtering TS packets, and therefore there are totally 4 sources can be chosen for TS out interface.

## **39.6.7 PVR**

PVR module provide you with the function to record the programs you want. The 4 sources can be assigned with PVR, and they are the same as TS out interface.

Assign the PVR length and PVR address, and then configure TSP\_PVR\_CTRL to start PVR module. If you want to stop PVR function during recording, write '1' to STOP bit (bit 0) to TSP\_PVR\_CTRL to stop it. Remember to take care of the status of PVR\_ON bit of TSP\_GFCG when programming the PVR-related registers.

## **39.6.8 PCR extraction**

PCR extraction can be enabled by configure PTIx\_PCRn\_CTRL. Then if the PID-matched TS data contain PCR field, the 33-bit PCR\_base field will be written corresponding PTIx\_PCRn\_H and PTIx\_PCRn\_L registers. An interrupt will be asserted if PCR interrupt is enabled.

Rockchip Confidential

## Chapter 40 High-Speed ADC Interface (HS-ADC)

### 40.1 Overview

HS-ADC Interface Unit is an interface unit for connecting the TS interface and GPS ADC interface to AMBA AHB bus. It fetches the bus data received by the TS interface and GPS ADC interface and stores them to internal asynchronous FIFO after the ADC clock is active. The HS-ADC Interface Unit generates the DMA request signal when data length of the asynchronous FIFO over the almost full level or almost empty level.

HS-ADC supports the following features:

- Support Transport-Stream(TS) Interface with 8bits data bus
- Support GPS interface with 2bits or 4bits data bus
- Support combined interrupt output, source including: full interrupt, empty interrupt
- Support DMA transfer mode through generating DMA request from the event of almost full or almost empty, etc.
- Support two channel mode: single channel and dual channel
- Support the most significant bit negation or not
- Support sign bit extension
- Support two storage mode: input data are stored to high 8bit or 10bit and stored to low 8bit or 10bit of 16bit when pushed into FIFO.
- Support an asynchronous build-in FIFO with 128x64 size

### 40.2 Block Diagram

The HS-ADC diagram is as follows.

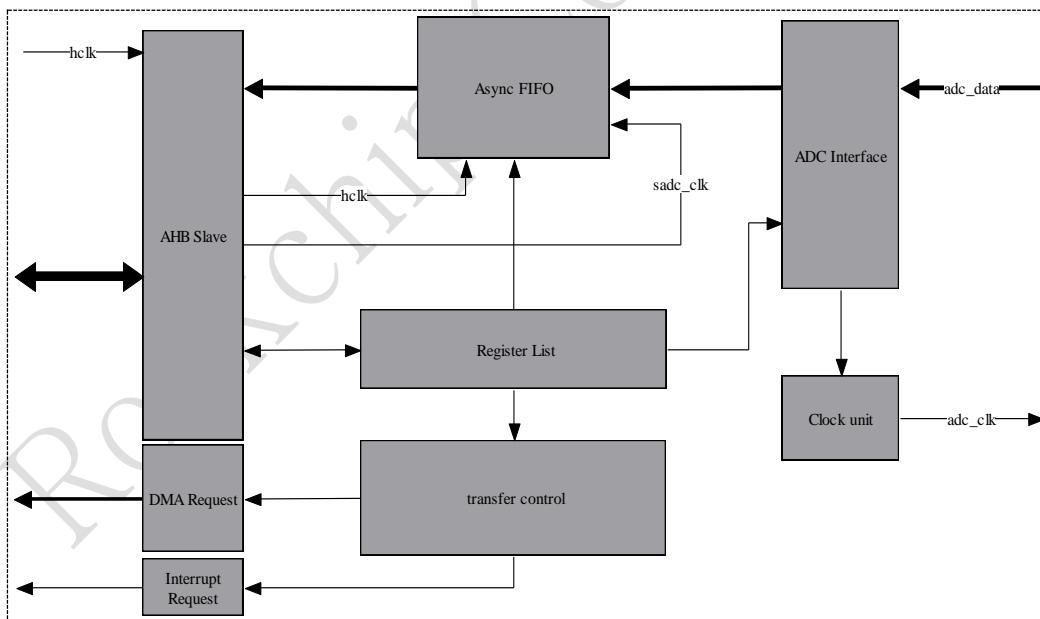


Fig. 40-1 HS-ADC Architecture

### 40.3 Function Description

This module can be configured for two interfaces: GPS interface, TS interface.

#### 40.3.1 GPS interface

When this module is used as GPS interface, user should configure GRF register to select 2bits or 4bitsGPS data input and gps\_clk as GPS clock input from pad.

Also, user should configure CRU\_CLKSEL22\_CON[5:4] to select gps\_clk as HS-ADC controller working clock source.

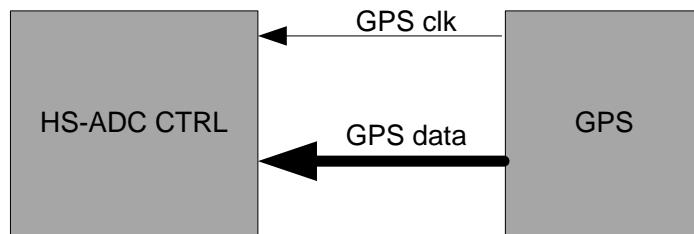


Fig. 40-2 GPS Application Diagram

#### 40.3.2 TS interface

When this module is used as TS interface, user should configure GRF register to select 8bit TS data, ts\_sync, ts\_valid and ts\_fail input and gps\_clk input as TS clock input from pad.

Also, user should configure CRU\_CLKSEL22\_CON[5:4] to select gps\_clk from pad as TS clock input.

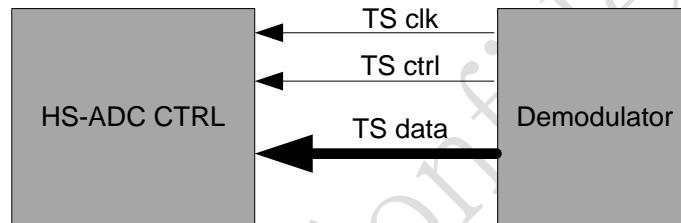


Fig. 40-3 TS Application Diagram

### 40.4 Register Description

#### 40.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
HSADC_CTRL	0x0000	W	0x00000000	Control register
HSADC_IER	0x0004	W	0x00000000	Interrupt control register
HSADC_ISR	0x0008	W	0x00000000	Interrupt status register
HSADC_TS_FAIL	0x000c	W	0x00000000	ts fail register
HSADC_CGCTL	0x0010	W	0x00000000	HSADC clock gating control
HSADC_DATA	0x0020	W	0x00000000	The data register of hsadc controller

Notes:**S**-Size: **B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**-WORD (32 bits) access

#### 40.4.2 Detail Register Description

##### HSADC\_CTRL

Address: Operational Base + offset (0x0000)

Control register

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:28	RO	0x0	reserved
27:24	RW	0x0	almost_full_level Define almost full trigger level 0x0~"0xf" - configure valid range (Notes: 1 level indicate 4 entries data in the async FIFO. and this configure range mapping to 64 - 124 entries data in the async FIFO.)
23:20	RO	0x0	reserved
19:16	RW	0x0	almost_empty_level Define almost empty trigger level 0x0~"0xf" - configure valid range (Notes: 1 level indicate 4 entries data in the async FIFO. and this configure range mapping to 0 - 60 entries data in the async FIFO.)
15	RW	0x0	gps_auto_gate_en GPS interface auto clock gating enable 1'b1: auto clock gating 1'b0: not auto clock gating
14	RW	0x0	ts_auto_gate_en TS interface auto clock gating enable 1'b1: auto clock gating 1'b0: not auto clock gating
13:12	RO	0x0	reserved
11	RW	0x0	gpsw GPS interface data width select 1'b0: 2bit data mode 1'b1: 4bit data mode
10	RW	0x0	ts_sync_en TS sync interface enable Field0000 Description
9	RW	0x0	ts_valid_en TS valid interface enable Enable ts interface "ts_valid" signal as data valid indicator 1'b0: disable 1'b1: enable
8	RW	0x0	ts_gps_sel MPEG-TS and GPS input select 1'b0: GPS is selected 1'b1: MPEG-TS is selected
7:6	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	dma_req_mode DMA request mode select 1'b1: almost full generate DMA request signal (Notes: this mode generate DMA request signal from almost full condition and cancel DMA request signal from almost empty condition. so you need configure two level by almost full level and almost empty level) 1'b0: almost empty generate DMA request signal (Notes: this mode generate DMA request signal from almost empty condition and that only once DMA request.)
4:1	RO	0x0	reserved
0	RW	0x0	adc_en HS-ADC Interface Unit Enable Bit 1'b1: enable (Notes: will return 1 when the hardware started transfer) 1'b0: disable (Notes: other bit can be modify only the hardware return 0)

**HSADC\_IER**

Address: Operational Base + offset (0x0004)

Interrupt control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	int_empty_en Interrupt en/disable bit for the empty interrupt flag of async FIFO 1'b1: enable 1'b0: disable
0	RW	0x0	int_full_en Interrupt en/disable bit for the full interrupt flag of async FIFO 1'b1: enable 1'b0: disable

**HSADC\_ISR**

Address: Operational Base + offset (0x0008)

Interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	int_empty_stat_ind Async FIFO empty interrupt flag 1'b1(R): This bit will be set to "1" when Async FIFO empty status and that only to read operation. 1'b0(W): Write "0" to bit for clear the interrupt flag and that only to write operation.
0	RW	0x0	int_full_stat_ind Async FIFO full interrupt flag 1'b1(R): This bit will be set to "1" when Async FIFO full status and that only to read operation. 1'b0(W): Write "0" to bit for clear the interrupt flag and that only to write operation.

**HSADC\_TS\_FAIL**

Address: Operational Base + offset (0x000c)  
ts fail register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	ts_fail_ahb TS stream fail indicator this signal only valid when select TS stream input( mpts=1) 1'b0: TS stream decode successfully 1'b1: TS stream decode fail

**HSADC(CGCTL)**

Address: Operational Base + offset (0x0010)  
HSADC Clock Gating control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RW	0x00000000	cycle_cfg clock gated cycles configuration when configure cg_enable to 1 and cycle_cfg to non-zero value,HSADC clock will be gated for cycle_cfg cycles ,then clock recover.
0	RW	0x0	cg_enable clock gating enable control 1'b0: clock gating disable 1'b1: clock gating enable

**HSADC\_DATA**

Address: Operational Base + offset (0x0020)  
Data register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	DATA

## 40.5 Interface Description

### 40.5.1 TS mode

Table 40-1 IOMUX configuration in TS mode

Module Pin	IO	Pad Name	IOMUX Setting
hsadc_data0	I	IO_CIFdata2_HOSTdin0_HSADCdata0_DVPgpio2a0	GPIO2A_IOMUX[1:0]= 2'b11
hsadc_data1	I	IO_CIFdata3_HOSTdin1_HSADCdata1_DVPgpio2a1	GPIO2A_IOMUX[3:2]= 2'b11
hsadc_data2	I	IO_CIFdata4_HOSTdin2_HSADCdata2_DVPgpio2a2	GPIO2A_IOMUX[5:4]= 2'b11
hsadc_data3	I	IO_CIFdata5_HOSTdin3_HSADCdata3_DVPgpio2a3	GPIO2A_IOMUX[7:6]= 2'b11
hsadc_data4	I	IO_CIFdata6_HOSTckinp_HSADCdata4_DVPgpio2a4	GPIO2A_IOMUX[9:8]= 2'b11
hsadc_data5	I	IO_CIFdata7_HOSTckinn_HSADCdata5_DVPgpio2a5	GPIO2A_IOMUX[11:10]= 2'b11
hsadc_data6	I	IO_CIFdata8_HOSTdin4_HSADCdata6_DVPgpio2a6	GPIO2A_IOMUX[13:12]= 2'b11
hsadc_data7	I	IO_CIFdata9_HOSTdin5_HSADCdata7_DVPgpio2a7	GPIO2A_IOMUX[15:14]= 2'b11
hsadc_valid	I	IO_CIFhref_HOSTdin7_HSADCTSvalid_DVPgpio2b1	GPIO2B_IOMUX[3:2]= 2'b11
hsadc_sync	I	IO_CIFVsync_HOSTdin6_HSADCTSsync_DVPgpio2b0	GPIO2B_IOMUX[1:0]= 2'b11
hsadc_err	I	IO_CIFclkout_HOSTTwkreq_HSADCTSfail_DVPgpio2b3	GPIO2B_IOMUX[7:6]= 2'b01
gps_clk	I	IO_CIFclkin_HOSTTwkack_GPSclk_HSADCclkout_DVPgpio2b2	GPIO2B_IOMUX[5:4]= 2'b11
gpst1_clk	I	IO_UART3GPSccts_GPSrfclk_GPST1clk_GPIO30gpio7b1	GPIO7B_IOMUX[3:2]= 2'b11

### 40.5.2 GPS mode

Table 40-2 IOMUX configuration in GPS mode

Module Pin	IO	Pad Name	IOMUX Setting
gps_clk	I	IO_CIFclkin_HOSTTwkack_GPSclk_HSADCclkout_DVPgpio2b2	GPIO2B_IOMUX[5:4]= 2'b11
gpst1_clk	I	IO_UART3GPSccts_GPSrfclk_GPST1clk_GPIO30gpio7b1	GPIO7B_IOMUX[3:2]= 2'b11
hsadc_data0	I	IO_CIFdata2_HOSTdin0_HSADCdata0_DVPgpio2a0	GPIO2A_IOMUX[1:0]= 2'b11
hsadc_data1	I	IO_CIFdata3_HOSTdin1_HSADCdata1_DVPgpio2a1	GPIO2A_IOMUX[3:2]= 2'b11
hsadc_data2	I	IO_CIFdata4_HOSTdin2_HSADCdata2_DVPgpio2a2	GPIO2A_IOMUX[5:4]= 2'b11
hsadc_data3	I	IO_CIFdata5_HOSTdin3_HSADCdata3_DVPgpio2a3	GPIO2A_IOMUX[7:6]= 2'b11

## 40.6 Application Notes

The following sections will describe the operation of DMA requests and DMA transfers.

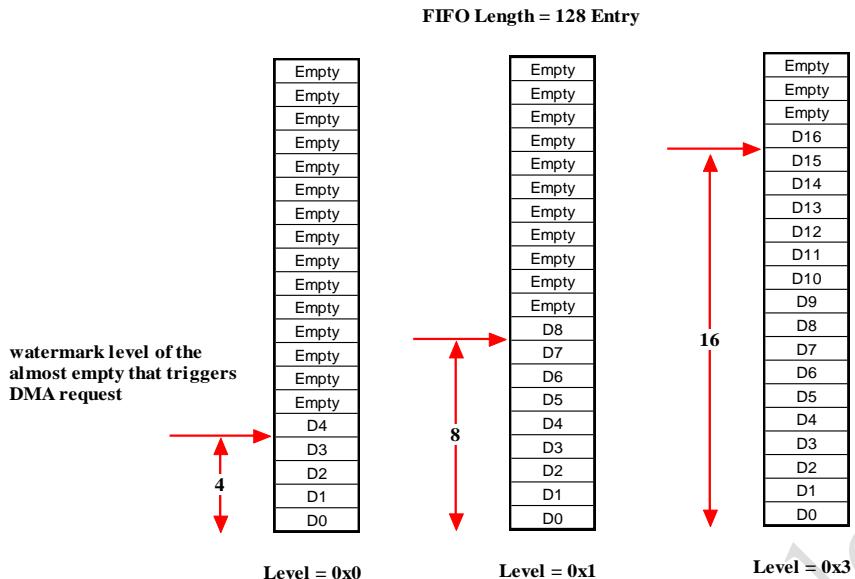


Fig. 40-4 Almost empty triggers a DMA request by DMA request mode

The DMA request signal will be generated from a watermark level trigger when data stored to FIFO over the watermark level of almost empty, where the watermark level can be configured through HSADC\_CTRL[19:16] by software. This DMA request mode doesn't care the watermark level of almost full. The sample for watermark level configuration is shown in figure above.

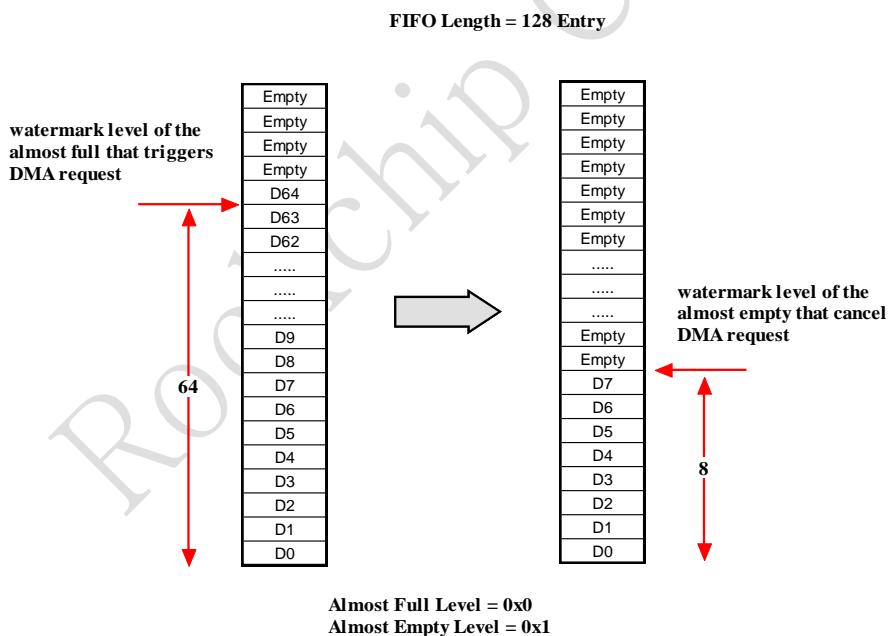


Fig. 40-5 Almost full triggers a DMA request by DMA request mode

The DMA request signal will be generated from a watermark level trigger when data stored to FIFO over the watermark level of almost full. It continues to generate request signal when the number of data in FIFO greater than watermark level of almost empty. This DMA request mode needs configure two watermark levels: watermark level of almost empty at the HSADC\_CTRL[19:16] and watermark level of almost full at the HSADC\_CTRL[27:24]. The sample for watermark level configuration is shown in figure above. When controller works in TS mode, the interface signal ts\_sync should always be used.

Rockchip Confidential

## Chapter 41 GMAC Ethernet Interface

### 41.1 Overview

The GMAC Ethernet Controller provides a complete Ethernet interface from processor to a Reduced Media Independent Interface (RMII) and Reduced Gigabit Media Independent Interface (RGMII) compliant Ethernet PHY.

The GMAC includes a DMA controller. The DMA controller efficiently moves packet data from microprocessor's RAM, formats the data for an IEEE 802.3-2002 compliant packet and transmits the data to an Ethernet Physical Interface (PHY). It also efficiently moves packet data from RXFIFO to microprocessor's RAM.

#### 41.1.1 Features

- Supports 10/100/1000-Mbps data transfer rates with the RGMII interfaces
- Supports 10/100-Mbps data transfer rates with the RMII interfaces
- Supports both full-duplex and half-duplex operation
  - Supports CSMA/CD Protocol for half-duplex operation
  - Supports packet bursting and frame extension in 1000 Mbps half-duplex operation
  - Supports IEEE 802.3x flow control for full-duplex operation
  - Optional forwarding of received pause control frames to the user application in full-duplex operation
  - Back-pressure support for half-duplex operation
  - Automatic transmission of zero-quanta pause frame on deassertion of flow control input in full-duplex operation
- Preamble and start-of-frame data (SFD) insertion in Transmit, and deletion in Receive paths
- Automatic CRC and pad generation controllable on a per-frame basis
- Options for Automatic Pad/CRC Stripping on receive frames
- Programmable frame length to support Standard Ethernet frames
- Programmable InterFrameGap (40-96 bit times in steps of 8)
- Supports a variety of flexible address filtering modes:
  - 64-bit Hash filter (optional) for multicast and uni-cast (DA) addresses
  - Option to pass all multicast addressed frames
  - Promiscuous mode support to pass all frames without any filtering for network monitoring
  - Passes all incoming packets (as per filter) with a status report
- Separate 32-bit status returned for transmission and reception packets
- Supports IEEE 802.1Q VLAN tag detection for reception frames
- MDIO Master interface for PHY device configuration and management
- Support detection of LAN wake-up frames and AMD Magic Packet frames
- Support checksum off-load for received IPv4 and TCP packets encapsulated by the Ethernet frame
- Support checking IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagrams
- Comprehensive status reporting for normal operation and transfers with errors
- Support per-frame Transmit/Receive complete interrupt control
- Supports 4-KB receive FIFO depths on reception.
- Supports 2-KB FIFO depth on transmission
- Automatic generation of PAUSE frame control or backpressure signal to the GMAC core based on Receive FIFO-fill (threshold configurable) level
- Handles automatic retransmission of Collision frames for transmission
- Discards frames on late collision, excessive collisions, excessive deferral and underrun conditions
- AXI interface to any CPU or memory
- Software can select the type of AXI burst (fixed and variable length burst) in the AXI Master interface
- Supports internal loopback on the RGMII/RMII for debugging
- Debug status register that gives status of FSMs in Transmit and Receive data-paths and FIFO fill-levels.

## 41.2 Block Diagram

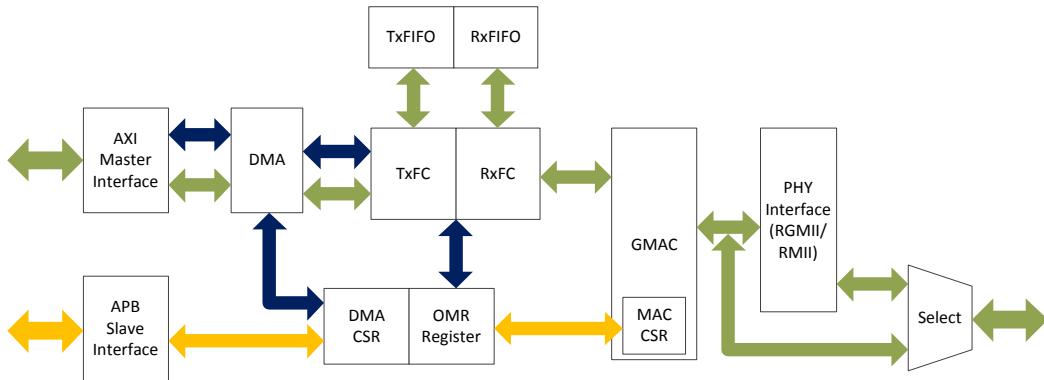


Fig. 41-1 GMAC architecture

The GMAC is broken up into multiple separate functional units. These blocks are interconnected in the MAC module. The block diagram shows the general flow of data and control signals between these blocks.

The GMAC transfers data to system memory through the AXI master interface. The host CPU uses the APB Slave interface to access the GMAC subsystem's control and status registers (CSRs).

The GMAC supports the PHY interfaces of reduced GMII (RGMII) and reduced MII (RMII).

The Transmit FIFO (Tx FIFO) buffers data read from system memory by the DMA before transmission by the GMAC Core. Similarly, the Receive FIFO (Rx FIFO) stores the Ethernet frames received from the line until they are transferred to system memory by the DMA. These are asynchronous FIFOs, as they also transfer the data between the application clock and the GMAC line clocks.

## 41.3 Function Description

### 41.3.1 Frame Structure

Data frames transmitted shall have the frame format shown in Fig 32-2.

<inter-frame><preamble><sfd><data><efd>

Fig. 41-2 MAC Frame structure

The preamble <preamble> begins a frame transmission. The bit value of the preamble field consists of 7 octets with the following bit values:

10101010 10101010 10101010 10101010 10101010 10101010 10101010

The SFD (start frame delimiter) <sfd> indicates the start of a frame and follows the preamble. The bit value is 10101011.

The data in a well formed frame shall consist of N octets data.

### 41.3.2 RMII Interface timing diagram

The Reduced Media Independent Interface (RMII) specification reduces the pin count between Ethernet PHYs and Switch ASICs (only in 10/100 mode). According to the IEEE 802.3u standard, an MII contains 16 pins for data and control. In devices incorporating multiple MAC or PHY interfaces (such as switches), the number of pins adds significant

cost with increase in port count. The RMII specification addresses this problem by reducing the pin count to 7 for each port - a 62.5% decrease in pin count.

The RMII module is instantiated between the GMAC and the PHY. This helps translation of the MAC's MII into the RMII. The RMII block has the following characteristics:

- Supports 10-Mbps and 100-Mbps operating rates. It does not support 1000-Mbps operation.
- Two clock references are sourced externally or CRU, providing independent, 2-bit wide transmit and receive paths.

### Transmit Bit Ordering

Each nibble from the MII must be transmitted on the RMII a di-bit at a time with the order of di-bit transmission shown in Fig.1-3. The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

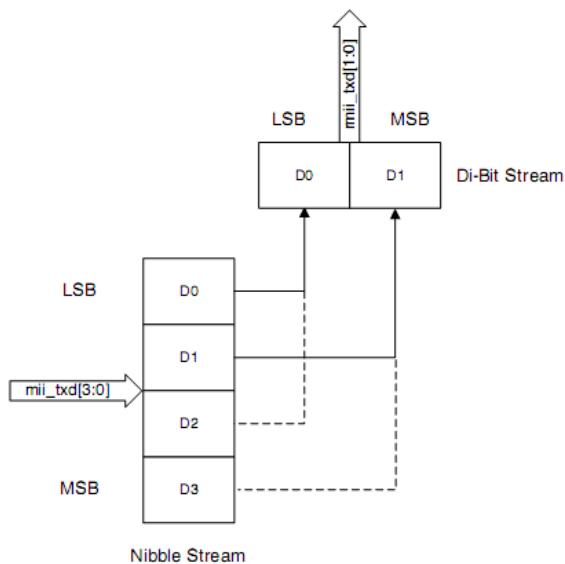


Fig. 41-3 RMII transmission bit ordering

### RMII Transmit Timing Diagrams

Fig.1-4 through 1-7 show MII-to-RMII transaction timing. The `clk_rmii_i` (REF\_CLK) frequency is 50MHz in RMII interface. In 10Mb/s mode, as the REF\_CLK frequency is 10 times as the data rate, the value on `rmii_txd_o[1:0]` (TXD[1:0]) shall be valid such that TXD[1:0] may be sampled every 10th cycle, regard-less of the starting cycle within the gRup and yield the correct frame data.

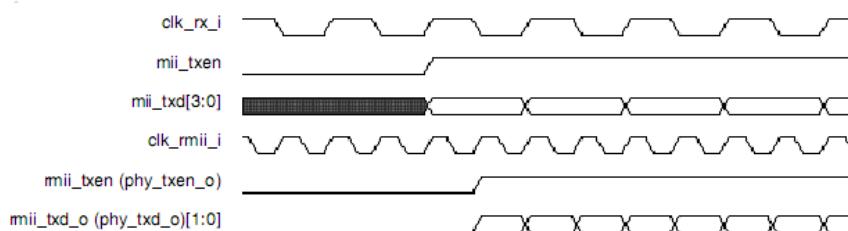


Fig. 41-4 Start of MII and RMII transmission in 100-Mbps mode

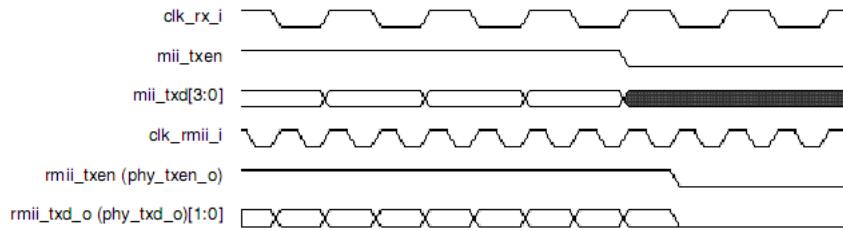


Fig. 41-5 End of MII and RMII Transmission in 100-Mbps Mode

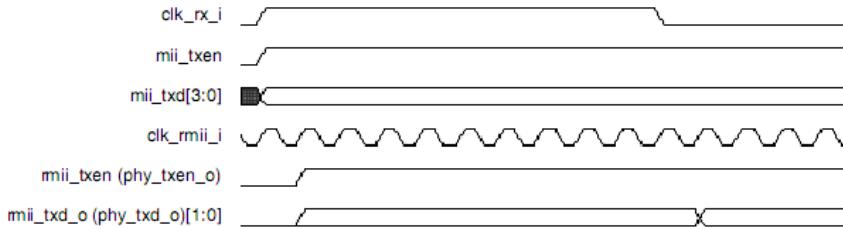


Fig. 41-6 Start of MII and RMII Transmission in 10-Mbps Mode

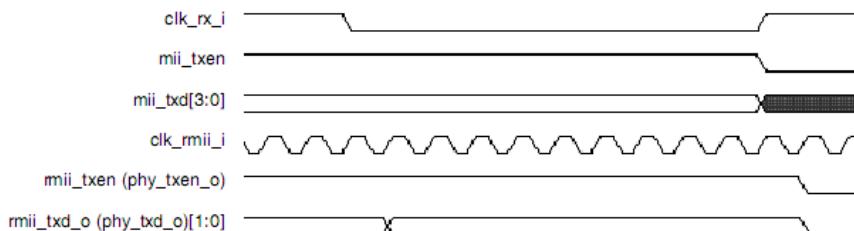


Fig. 41-7 End of MII and RMII Transmission in 10-Mbps Mode

## Receive Bit Ordering

Each nibble is transmitted to the MII from the di-bit received from the RMII in the nibble transmission order shown in Fig. 1-8. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

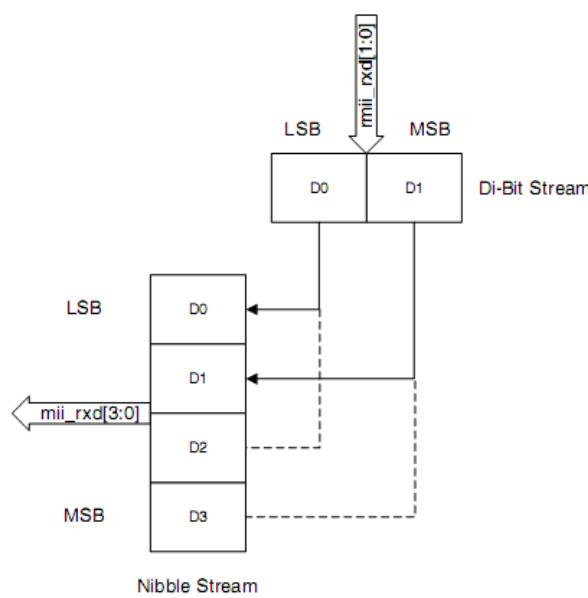


Fig. 41-8 RMII receive bit ordering

### 41.3.3 RGMII interface

The Reduced Gigabit Media Independent Interface (RGMII) specification reduces the pin count of the interconnection between the GMAC 10/100/1000 controller and the PHY for GMII and MII interfaces. To achieve this, the data path and control signals are reduced and multiplexed together with both the edges of the transmit and receive clocks. For gigabit operation the clocks operate at 125 MHz; for 10/100 operation, the clock rates are 2.5 MHz/25 MHz.

In the GMAC 10/100/1000 controller, the RGMII module is instantiated between the GMAC core's GMII and the PHY to translate the control and data signals between the GMII and RGMII protocols.

The RGMII block has the following characteristics:

- Supports 10-Mbps, 100-Mbps, and 1000-Mbps operation rates.
- For the RGMII block, no extra clock is required because both the edges of the incoming clocks are used.
- The RGMII block extracts the in-band (link speed, duplex mode and link status) status signals from the PHY and provides them to the GMAC core logic for link detection.

### 41.3.4 Management Interface

The MAC management interface provides a simple, two-wire, serial interface to connect the GMAC and a managed PHY, for the purposes of controlling the PHY and gathering status from the PHY. The management interface consists of a pair of signals that transport the management information across the MII bus: MDIO and MDC.

The GMAC initiates the management write/read operation. The clock gmii\_mdc\_o(MDC) is a divided clock from the application clock pclk\_gmac. The divide factor depends on the clock range setting in the GMII address register. Clock range is set as follows:

Selection	pclk_gmac	MDC Clock
0000	60-100 MHz	pclk_gmac/42
0001	100-150 MHz	pclk_gmac/62
0010	20-35 MHz	pclk_gmac/16
0011	35-60 MHz	pclk_gmac/26
0100	150-250 MHz	pclk_gmac/102
0101	250-300 MHz	pclk_gmac/124
0110, 0111	Reserved	

The MDC is the derivative of the application clock pclk\_gmac. The management operation is performed through the gmii\_mdio\_i, gmii\_mdo\_o and gmii\_mdo\_o\_e signals. A three-state buffer is implemented in the PAD.

The frame structure on the MDIO line is shown below.

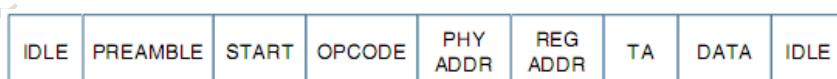


Fig. 41-9 MDIO frame structure

- IDLE: The mdio line is three-state; there is no clock on gmii\_mdc\_o
- PREAMBLE: 32 continuous bits of value 1
- START: Start-of-frame is 2'b01
- OPCODE: 2'b10 for read and 2'b01 for write
- PHY ADDR: 5-bit address select for one of 32 PHYs
- REG ADDR: Register address in the selected PHY
- TA: Turnaround is 2'bZ0 for read and 2'b10 for Write
- DATA: Any 16-bit value. In a write operation, the GMAC drives mdio; in a read operation, PHY drives it.

### 41.3.5 Power Management Block

Power management(PMT) supports the reception of network (remote) wake-up frames and Magic Packet frames. PMT does not perform the clock gate function, but generates interrupts for wake-up frames and Magic Packets received by the GMAC. The PMT block sits on the receiver path of the GMAC and is enabled with remote wake-up frame enable and Magic Packet enable. These enables are in the PMT control and status register and are programmed by the application.

When the power down mode is enabled in the PMT, then all received frames are dropped by the core and they are not forwarded to the application. The core comes out of the power down mode only when either a Magic Packet or a Remote Wake-up frame is received and the corresponding detection is enabled.

#### Remote Wake-Up Frame Detection

When the GMAC is in sleep mode and the remote wake-up bit is enabled in register GMAC\_PMT\_CTRL\_STA (0x002C), normal operation is resumed after receiving a remote wake-up frame. The application writes all eight wake-up filter registers, by performing a sequential write to address (0028). The application enables remote wake-up by writing a 1 to bit 2 of the register GMAC\_PMT\_CTRL\_STA.

PMT supports four programmable filters that allow support of different receive frame patterns. If the incoming frame passes the address filtering of Filter Command, and if Filter CRC-16 matches the incoming examined pattern, then the wake-up frame is received.

Filter\_offset (minimum value 12, which refers to the 13th byte of the frame) determines the offset from which the frame is to be examined. Filter Byte Mask determines which bytes of the frame must be examined. The thirty-first bit of Byte Mask must be set to zero.

The remote wake-up CRC block determines the CRC value that is compared with Filter CRC-16. The wake-up frame is checked only for length error, FCS error, dribble bit error, GMII error, collision, and to ensure that it is not a runt frame. Even if the wake-up frame is more than 512 bytes long, if the frame has a valid CRC value, it is considered valid. Wake-up frame detection is updated in the register GMAC\_PMT\_CTRL\_STA for every remote Wake-up frame received. A PMT interrupt to the application triggers a read to the GMAC\_PMT\_CTRL\_STA register to determine reception of a wake-up frame.

#### Magic Packet Detection

The Magic Packet frame is based on a method that uses Advanced Micro Device's Magic Packet technology to power up the sleeping device on the network. The GMAC receives a specific packet of information, called a Magic Packet, addressed to the node on the network.

Only Magic Packets that are addressed to the device or a broadcast address will be checked to determine whether they meet the wake-up requirements. Magic Packets that pass the address filtering (unicast or broadcast) will be checked to determine whether they meet the remote Wake-on-LAN data format of 6 bytes of all ones followed by a GMAC Address appearing 16 times.

The application enables Magic Packet wake-up by writing a 1 to Bit 1 of the register GMAC\_PMT\_CTRL\_STA. The PMT block constantly monitors each frame addressed to the node for a specific Magic Packet pattern. Each frame received is checked for a 48'hFF\_FF\_FF\_FF\_FF pattern following the destination and source address field. The PMT block then checks the frame for 16 repetitions of the GMAC address without any breaks or interruptions. In case of a break in the 16 repetitions of the address, the 48'hFF\_FF\_FF\_FF\_FF pattern is scanned for again in the incoming frame. The 16 repetitions can be anywhere in the frame, but must be preceded by the synchronization stream (48'hFF\_FF\_FF\_FF\_FF). The device will also accept a multicast frame, as long as the 16 duplications of the GMAC address are detected.

If the MAC address of a node is 48'h00\_11\_22\_33\_44\_55, then the GMAC scans for the data sequence:

Destination Address Source Address ..... FF FF FF FF FF FF  
 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55  
 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55  
 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55  
 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55  
 ...CRC

Magic Packet detection is updated in the PMT Control and Status register for Magic Packet received. A PMT interrupt to the Application triggers a read to the PMT CSR to determine whether a Magic Packet frame has been received.

### 41.3.6 MAC Management Counters

The counters in the MAC Management Counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted frames. These include a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the MAC Control Interface (MCI). Non-32-bit accesses are allowed as long as the address is word-aligned.

The organization of these registers is shown in Register Description. The MMCs are accessed using transactions, in the same way the CSR address space is accessed. The Register Description in this chapter describe the various counters and list the address for each of the statistics counters. This address will be used for Read/Write accesses to the desired transmit/receive counter.

The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet frames.

## 41.4 Register description

### 41.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
GMAC_MAC_CONF	0x0000	W	0x00000000	MAC Configuration Register
GMAC_MAC_FRM_FILT	0x0004	W	0x00000000	MAC Frame Filter
GMAC_HASH_TAB_HI	0x0008	W	0x00000000	Hash Table High Register
GMAC_HASH_TAB_LO	0x000c	W	0x00000000	Hash Table Low Register
GMAC_GMII_ADDR	0x0010	W	0x00000000	GMII Address Register
GMAC_GMII_DATA	0x0014	W	0x00000000	GMII Data Register
GMAC_FLOW_CTRL	0x0018	W	0x00000000	Flow Control Register
GMAC_VLAN_TAG	0x001c	W	0x00000000	VLAN Tag Register
GMAC_DEBUG	0x0024	W	0x00000000	Debug register
GMAC_PMT_CTRL_STA	0x002c	W	0x00000000	PMT Control and Status Register

Name	Offset	Size	Reset Value	Description
GMAC_INT_STATUS	0x0038	W	0x00000000	Interrupt Status Register
GMAC_INT_MASK	0x003c	W	0x00000000	Interrupt Mask Register
GMAC_MAC_ADDR0_HI	0x0040	W	0x0000ffff	MAC Address0 High Register
GMAC_MAC_ADDR0_LO	0x0044	W	0xffffffff	MAC Address0 Low Register
GMAC_AN_CTRL	0x00c0	W	0x00000000	AN Control Register
GMAC_AN_STATUS	0x00c4	W	0x00000008	AN Status Register
GMAC_AN_ADV	0x00c8	W	0x000001e0	Auto_Negotiation Advertisement Register
GMAC_AN_LINK_PAR_T_AB	0x00cc	W	0x00000000	Auto_Negotiation Link Partner Ability Register
GMAC_AN_EXP	0x00d0	W	0x00000000	Auto_Negotiation Expansion Register
GMAC_INTF_MODE_STA	0x00d8	W	0x00000000	RGMII Status Register
GMAC_MMC_CTRL	0x0100	W	0x00000000	MMC Control Register
GMAC_MMC_RX_INTERRUPT	0x0104	W	0x00000000	MMC Receive Interrupt Register
GMAC_MMC_TX_INTERRUPT	0x0108	W	0x00000000	MMC Transmit Interrupt Register
GMAC_MMC_RX_INTERRUPT_MSK	0x010c	W	0x00000000	MMC Receive Interrupt Mask Register
GMAC_MMC_TX_INTERRUPT_MSK	0x0110	W	0x00000000	MMC Transmit Interrupt Mask Register
GMAC_MMC_TXOCTET_TCNT_GB	0x0114	W	0x00000000	MMC TX OCTET Good and Bad Counter
GMAC_MMC_TXFRMCNT_GB	0x0118	W	0x00000000	MMC TX Frame Good and Bad Counter
GMAC_MMC_TXUNDFLWERR	0x0148	W	0x00000000	MMC TX Underflow Error
GMAC_MMC_TXCARERR	0x0160	W	0x00000000	MMC TX Carrier Error
GMAC_MMC_TXOCTET_TCNT_G	0x0164	W	0x00000000	MMC TX OCTET Good Counter
GMAC_MMC_TXFRMCNT_G	0x0168	W	0x00000000	MMC TX Frame Good Counter
GMAC_MMC_RXFRMCNT_GB	0x0180	W	0x00000000	MMC RX Frame Good and Bad Counter
GMAC_MMC_RXOCTET_TCNT_GB	0x0184	W	0x00000000	MMC RX OCTET Good and Bad Counter
GMAC_MMC_RXOCTET_TCNT_G	0x0188	W	0x00000000	MMC RX OCTET Good Counter

Name	Offset	Size	Reset Value	Description
GMAC_MMCRXMCFRMCNT_G	0x0190	W	0x00000000	MMC RX Multicast Frame Good Counter
GMAC_MMCRXCRCERRR	0x0194	W	0x00000000	MMC RX Carrier
GMAC_MMCRXLENERRR	0x01c8	W	0x00000000	MMC RX Length Error
GMAC_MMCRXFIFOVRFLW	0x01d4	W	0x00000000	MMC RX FIFO Overflow
GMAC_MMCIIPCINT_MSK	0x0200	W	0x00000000	MMC Receive Checksum Offload Interrupt Mask Register
GMAC_MMCIINTR	0x0208	W	0x00000000	MMC Receive Checksum Offload Interrupt Register
GMAC_MMCRXIPV4GFRM	0x0210	W	0x00000000	MMC RX IPV4 Good Frame
GMAC_MMCRXIPV4HDERRFRM	0x0214	W	0x00000000	MMC RX IPV4 Head Error Frame
GMAC_MMCRXIPV6GFRM	0x0224	W	0x00000000	MMC RX IPV6 Good Frame
GMAC_MMCRXIPV6HDERRFRM	0x0228	W	0x00000000	MMC RX IPV6 Head Error Frame
GMAC_MMCRXUDPRERFRM	0x0234	W	0x00000000	MMC RX UDP Error Frame
GMAC_MMCRXTCPERRFRM	0x023c	W	0x00000000	MMC RX TCP Error Frame
GMAC_MMCRXICMPERRFRM	0x0244	W	0x00000000	MMC RX ICMP Error Frame
GMAC_MMCRXIPV4HDERROCT	0x0254	W	0x00000000	MMC RX OCTET IPV4 Head Error
GMAC_MMCRXIPV6HDERROCT	0x0268	W	0x00000000	MMC RX OCTET IPV6 Head Error
GMAC_MMCRXUDPRERROCT	0x0274	W	0x00000000	MMC RX OCTET UDP Error
GMAC_MMCRXTCPERRROCT	0x027c	W	0x00000000	MMC RX OCTET TCP Error
GMAC_MMCRXICMPERRROCT	0x0284	W	0x00000000	MMC RX OCTET ICMP Error
GMAC_BUS_MODE	0x1000	W	0x00020101	Bus Mode Register
GMAC_TX_POLL_DEMAND	0x1004	W	0x00000000	Transmit Poll Demand Register
GMAC_RX_POLL_DEMAND	0x1008	W	0x00000000	Receive Poll Demand Register
GMAC_RX_DESC_LIST_ADDR	0x100c	W	0x00000000	Receive Descriptor List Address Register

Name	Offset	Size	Reset Value	Description
GMAC_TX_DESC_LIST_ADDR	0x1010	W	0x00000000	Transmit Descriptor List Address Register
GMAC_STATUS	0x1014	W	0x00000000	Status Register
GMAC_OP_MODE	0x1018	W	0x00000000	Operation Mode Register
GMAC_INT_ENA	0x101c	W	0x00000000	Interrupt Enable Register
GMAC_OVERFLOW_CNT	0x1020	W	0x00000000	Missed Frame and Buffer Overflow Counter Register
GMAC_REC_INT_WDT_TIMER	0x1024	W	0x00000000	Receive Interrupt Watchdog Timer Register
GMAC_AXI_BUS_MODE	0x1028	W	0x00110001	AXI Bus Mode Register
GMAC_AXI_STATUS	0x102c	W	0x00000000	AXI Status Register
GMAC_CUR_HOST_TX_DESC	0x1048	W	0x00000000	Current Host Transmit Descriptor Register
GMAC_CUR_HOST_RX_DESC	0x104c	W	0x00000000	Current Host Receive Descriptor Register
GMAC_CUR_HOST_TX_Buf_ADDR	0x1050	W	0x00000000	Current Host Transmit Buffer Address Register
GMAC_CUR_HOST_RX_Buf_ADDR	0x1054	W	0x00000000	Current Host Receive Buffer Address Register
GMAC_HW_FEA_REG	0x1058	W	0x000d0f17	The presence of the optional features/functions of the core

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

#### 41.4.2 Detail Register Description

##### GMAC\_MAC\_CONF

Address: Operational Base + offset (0x0000)

MAC Configuration Register

Bit	Attr	Reset Value	Description
31:25	RO	0x0	reserved
24	RW	0x0	TC Transmit Configuration in RGMII/SGMII/SMII When set, this bit enables the transmission of duplex mode, link speed, and link up/down information to the PHY in the RGMII ports. When this bit is reset, no such information is driven to the PHY.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
23	RW	0x0	<p>WD Watchdog Disable When this bit is set, the GMAC disables the watchdog timer on the receiver, and can receive frames of up to 16,384 bytes. When this bit is reset, the GMAC allows no more than 2,048 bytes (10,240 if JE is set high) of the frame being received and cuts off any bytes received after that.</p>
22	RW	0x0	<p>JD Jabber Disable When this bit is set, the GMAC disables the jabber timer on the transmitter, and can transfer frames of up to 16,384 bytes. When this bit is reset, the GMAC cuts off the transmitter if the application sends out more than 2,048 bytes of data (10,240 if JE is set high) during transmission.</p>
21	RW	0x0	<p>BE Frame Burst Enable When this bit is set, the GMAC allows frame bursting during transmission in GMII Half-Duplex mode.</p>
20	RO	0x0	reserved
19:17	RW	0x0	<p>IFG Inter-Frame Gap These bits control the minimum IFG between frames during transmission. 3'b000: 96 bit times 3'b001: 88 bit times 3'b010: 80 bit times ... 3'b111: 40 bit times</p>
16	RW	0x0	<p>DCRS Disable Carrier Sense During Transmission When set high, this bit makes the MAC transmitter ignore the (G)MII CRS signal during frame transmission in Half-Duplex mode. This request results in no errors generated due to Loss of Carrier or No Carrier during such transmission. When this bit is low, the MAC transmitter generates such errors due to Carrier Sense and will even abort the transmissions.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RW	0x0	PS Port Select Selects between GMII and MII: 1'b0: GMII (1000 Mbps) 1'b1: MII (10/100 Mbps)
14	RW	0x0	FES Speed Indicates the speed in Fast Ethernet (MII) mode: 1'b0: 10 Mbps 1'b1: 100 Mbps
13	RW	0x0	DO Disable Receive Own When this bit is set, the GMAC disables the reception of frames when the gmii_txen_o is asserted in Half-Duplex mode. When this bit is reset, the GMAC receives all packets that are given by the PHY while transmitting.
12	RW	0x0	LM Loopback Mode When this bit is set, the GMAC operates in loopback mode at GMII/MII. The (G)MII Receive clock input (clk_rx_i) is required for the loopback to work properly, as the Transmit clock is not looped-back internally.
11	RW	0x0	DM Duplex Mode When this bit is set, the GMAC operates in a Full-Duplex mode where it can transmit and receive simultaneously. This bit is RO with default value of 1'b1 in Full-Duplex-only configuration.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
10	RW	0x0	<p>IPC Checksum Offload When this bit is set, the GMAC calculates the 16-bit one's complement of the one's complement sum of all received Ethernet frame payloads. It also checks whether the IPv4 Header checksum (assumed to be bytes 25-26 or 29-30 (VLAN-tagged) of the received Ethernet frame) is correct for the received frame and gives the status in the receive status word. The GMAC core also appends the 16-bit checksum calculated for the IP header datagram payload (bytes after the IPv4 header) and appends it to the Ethernet frame transferred to the application (when Type 2 COE is deselected). When this bit is reset, this function is disabled. When Type 2 COE is selected, this bit, when set, enables IPv4 checksum checking for received frame payloads TCP/UDP/ICMP headers. When this bit is reset, the COE function in the receiver is disabled and the corresponding PCE and IP HCE status bits are always cleared.</p>
9	RW	0x0	<p>DR Disable Retry When this bit is set, the GMAC will attempt only 1 transmission. When a collision occurs on the GMII/MII, the GMAC will ignore the current frame transmission and report a Frame Abort with excessive collision error in the transmit frame status. When this bit is reset, the GMAC will attempt retries based on the settings of BL.</p>
8	RW	0x0	<p>LUD Link Up/Down Indicates whether the link is up or down during the transmission of configuration in RGMII interface: 1'b0: Link Down 1'b1: Link Up</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	<p>ACS Automatic Pad/CRC Stripping When this bit is set, the GMAC strips the Pad/FCS field on incoming frames only if the length's field value is less than or equal to 1,500 bytes. All received frames with length field greater than or equal to 1,501 bytes are passed to the application without stripping the Pad/FCS field.</p> <p>When this bit is reset, the GMAC will pass all incoming frames to the Host unmodified.</p>
6:5	RW	0x0	<p>BL Back-Off Limit The Back-Off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps and 512 bit times for 10/100 Mbps) the GMAC waits before rescheduling a transmission attempt during retries after a collision. This bit is applicable only to Half-Duplex mode and is reserved (RO) in Full-Duplex-only configuration.</p> <p>2'b00: k = min (n, 10) 2'b01: k = min (n, 8) 2'b10: k = min (n, 4) 2'b11: k = min (n, 1), where n = retransmission attempt. The random integer r takes the value in the range 0 = r &lt; 2^k</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RW	0x0	<p>DC Deferral Check When this bit is set, the deferral check function is enabled in the GMAC. The GMAC will issue a Frame Abort status, along with the excessive deferral error bit set in the transmit frame status when the transmit state machine is deferred for more than 24,288 bit times in 10/100-Mbps mode. If the Core is configured for 1000 Mbps operation, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but is prevented because of an active CRS (carrier sense) signal on the GMII/MII. Defer time is not cumulative. If the transmitter defers for 10,000 bit times, then transmits, collides, backs off, and then has to defer again after completion of back-off, the deferral timer resets to 0 and restarts.</p> <p>When this bit is reset, the deferral check function is disabled and the GMAC defers until the CRS signal goes inactive.</p>
3	RW	0x0	<p>TE Transmitter Enable When this bit is set, the transmit state machine of the GMAC is enabled for transmission on the GMII/MII. When this bit is reset, the GMAC transmit state machine is disabled after the completion of the transmission of the current frame, and will not transmit any further frames.</p>
2	RW	0x0	<p>RE Receiver Enable When this bit is set, the receiver state machine of the GMAC is enabled for receiving frames from the GMII/MII. When this bit is reset, the GMAC receive state machine is disabled after the completion of the reception of the current frame, and will not receive any further frames from the GMII/MII.</p>
1:0	RO	0x0	reserved

**GMAC\_MAC\_FRM\_FILT**

Address: Operational Base + offset (0x0004)  
MAC Frame Filter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	<p>RA Receive All</p> <p>When this bit is set, the GMAC Receiver module passes to the Application all frames received irrespective of whether they pass the address filter. The result of the SA/DA filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word. When this bit is reset, the Receiver module passes to the Application only those frames that pass the SA/DA address filter.</p>
30:11	RO	0x0	reserved
10	RW	0x0	<p>HPF Hash or Perfect Filter</p> <p>When set, this bit configures the address filter to pass a frame if it matches either the perfect filtering or the hash filtering as set by HMC or HUC bits. When low and if the HUC/HMC bit is set, the frame is passed only if it matches the Hash filter.</p>
9	RW	0x0	<p>SAF Source Address Filter Enable</p> <p>The GMAC core compares the SA field of the received frames with the values programmed in the enabled SA registers. If the comparison matches, then the SAMatch bit of RxStatus Word is set high. When this bit is set high and the SA filter fails, the GMAC drops the frame. When this bit is reset, then the GMAC Core forwards the received frame to the application and with the updated SA Match bit of the RxStatus depending on the SA address comparison.</p>
8	RW	0x0	<p>SAIF SA Inverse Filtering</p> <p>When this bit is set, the Address Check block operates in inverse filtering mode for the SA address comparison. The frames whose SA matches the SA registers will be marked as failing the SA Address filter.</p> <p>When this bit is reset, frames whose SA does not match the SA registers will be marked as failing the SA Address filter.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:6	RW	0x0	<p>PCF Pass Control Frames These bits control the forwarding of all control frames (including unicast and multicast PAUSE frames). Note that the processing of PAUSE control frames depends only on RFE of Register GMAC_FLOW_CTRL[2].</p> <p>2'b00: GMAC filters all control frames from reaching the application.</p> <p>2'b01: GMAC forwards all control frames except PAUSE control frames to application even if they fail the Address filter.</p> <p>2'b10: GMAC forwards all control frames to application even if they fail the Address Filter.</p> <p>2'b11: GMAC forwards control frames that pass the Address Filter.</p>
5	RW	0x0	<p>DBF Disable Broadcast Frames When this bit is set, the AFM module filters all incoming broadcast frames.</p> <p>When this bit is reset, the AFM module passes all received broadcast frames.</p>
4	RW	0x0	<p>PM Pass All Multicast When set, this bit indicates that all received frames with a multicast destination address (first bit in the destination address field is '1') are passed.</p> <p>When reset, filtering of multicast frame depends on HMC bit.</p>
3	RW	0x0	<p>DAIF DA Inverse Filtering When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast frames.</p> <p>When reset, normal filtering of frames is performed.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x0	HMC Hash Multicast When set, MAC performs destination address filtering of received multicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for multicast frames, that is, it compares the DA field with the values programmed in DA registers.
1	RW	0x0	HUC Hash Unicast When set, MAC performs destination address filtering of unicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for unicast frames, that is, it compares the DA field with the values programmed in DA registers.
0	RW	0x0	PR Promiscuous Mode When this bit is set, the Address Filter module passes all incoming frames regardless of its destination or source address. The SA/DA Filter Fails status bits of the Receive Status Word will always be cleared when PR is set.

**GMAC\_HASH\_TAB\_HI**

Address: Operational Base + offset (0x0008)

Hash Table High Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	HTH Hash Table High This field contains the upper 32 bits of Hash table

**GMAC\_HASH\_TAB\_LO**

Address: Operational Base + offset (0x000c)

Hash Table Low Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	HTL Hash Table Low This field contains the lower 32 bits of Hash table

**GMAC\_GMII\_ADDR**

Address: Operational Base + offset (0x0010)

GMII Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:11	RW	0x00	PA Physical Layer Address This field tells which of the 32 possible PHY devices are being accessed
10:6	RW	0x00	GR GMII Register These bits select the desired GMII register in the selected PHY device

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>																																				
5:2	RW	0x0	<p>CR APB Clock Range The APB Clock Range selection determines the frequency of the MDC clock as per the pclk_gmac frequency used in your design. The suggested range of pclk_gmac frequency applicable for each value below (when Bit[5] = 0) ensures that the MDC clock is approximately between the frequency range 1.0 MHz - 2.5 MHz.</p> <table> <thead> <tr> <th>Selection</th> <th>MDC Clock</th> </tr> </thead> <tbody> <tr> <td>pclk_gmac</td> <td>60-100 MHz</td> </tr> <tr> <td>pclk_gmac/42</td> <td>100-150 MHz</td> </tr> <tr> <td>pclk_gmac/62</td> <td>20-35 MHz</td> </tr> <tr> <td>pclk_gmac/16</td> <td>35-60 MHz</td> </tr> <tr> <td>pclk_gmac/26</td> <td>150-250 MHz</td> </tr> <tr> <td>pclk_gmac/102</td> <td>250-300 MHz</td> </tr> <tr> <td>pclk_gmac/124</td> <td></td> </tr> <tr> <td>0110, 0111</td> <td>Reserved</td> </tr> </tbody> </table> <p>When bit 5 is set, you can achieve MDC clock of frequency higher than the IEEE 802.3 specified frequency limit of 2.5 MHz and program a clock divider of lower value. For example, when pclk_gmac is of frequency 100 Mhz and you program these bits as "1010", then the resultant MDC clock will be of 12.5 Mhz which is outside the limit of IEEE 802.3 specified range. Please program the values given below only if the interfacing chips supports faster MDC clocks.</p> <table> <thead> <tr> <th>Selection</th> <th>MDC Clock</th> </tr> </thead> <tbody> <tr> <td>1000</td> <td>pclk_gmac/4</td> </tr> <tr> <td>1001</td> <td>pclk_gmac/6</td> </tr> <tr> <td>1010</td> <td>pclk_gmac/8</td> </tr> <tr> <td>1011</td> <td>pclk_gmac/10</td> </tr> <tr> <td>1100</td> <td>pclk_gmac/12</td> </tr> <tr> <td>1101</td> <td>pclk_gmac/14</td> </tr> <tr> <td>1110</td> <td>pclk_gmac/16</td> </tr> <tr> <td>1111</td> <td>pclk_gmac/18</td> </tr> </tbody> </table>	Selection	MDC Clock	pclk_gmac	60-100 MHz	pclk_gmac/42	100-150 MHz	pclk_gmac/62	20-35 MHz	pclk_gmac/16	35-60 MHz	pclk_gmac/26	150-250 MHz	pclk_gmac/102	250-300 MHz	pclk_gmac/124		0110, 0111	Reserved	Selection	MDC Clock	1000	pclk_gmac/4	1001	pclk_gmac/6	1010	pclk_gmac/8	1011	pclk_gmac/10	1100	pclk_gmac/12	1101	pclk_gmac/14	1110	pclk_gmac/16	1111	pclk_gmac/18
Selection	MDC Clock																																						
pclk_gmac	60-100 MHz																																						
pclk_gmac/42	100-150 MHz																																						
pclk_gmac/62	20-35 MHz																																						
pclk_gmac/16	35-60 MHz																																						
pclk_gmac/26	150-250 MHz																																						
pclk_gmac/102	250-300 MHz																																						
pclk_gmac/124																																							
0110, 0111	Reserved																																						
Selection	MDC Clock																																						
1000	pclk_gmac/4																																						
1001	pclk_gmac/6																																						
1010	pclk_gmac/8																																						
1011	pclk_gmac/10																																						
1100	pclk_gmac/12																																						
1101	pclk_gmac/14																																						
1110	pclk_gmac/16																																						
1111	pclk_gmac/18																																						

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	GW GMII Write When set, this bit tells the PHY that this will be a Write operation using register GMAC_GMII_DATA. If this bit is not set, this will be a Read operation, placing the data in register GMAC_GMII_DATA.
0	W1C	0x0	GB GMII Busy This bit should read a logic 0 before writing to Register GMII_ADDR and Register GMII_DATA. This bit must also be set to 0 during a Write to Register GMII_ADDR. During a PHY register access, this bit will be set to 1'b1 by the Application to indicate that a Read or Write access is in progress. Register GMII_DATA (GMII Data) should be kept valid until this bit is cleared by the GMAC during a PHY Write operation. The Register GMII_DATA is invalid until this bit is cleared by the GMAC during a PHY Read operation. The Register GMII_ADDR (GMII Address) should not be written to until this bit is cleared.

**GMAC\_GMII\_DATA**

Address: Operational Base + offset (0x0014)

GMII Data Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	GD GMII Data This contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.

**GMAC\_FLOW\_CTRL**

Address: Operational Base + offset (0x0018)

Flow Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>										
31:16	RW	0x0000	<p>PT Pause Time This field holds the value to be used in the Pause Time field in the transmit control frame. If the Pause Time bits is configured to be double-synchronized to the (G)MII clock domain, then consecutive writes to this register should be performed only after at least 4 clock cycles in the destination clock domain.</p>										
15:8	RO	0x0	reserved										
7	RW	0x0	<p>DZPQ Disable Zero-Quanta Pause When set, this bit disables the automatic generation of Zero-Quanta Pause Control frames on the deassertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i/mti_flowctrl_i). When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled.</p>										
6	RO	0x0	reserved										
5:4	RW	0x0	<p>PLT Pause Low Threshold This field configures the threshold of the PAUSE timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of PAUSE Frame. The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot-times), and PLT = 01, then a second PAUSE frame is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot-times after the first PAUSE frame is transmitted.</p> <table> <thead> <tr> <th>Selection</th> <th>Threshold</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>Pause time minus 4 slot times</td> </tr> <tr> <td>01</td> <td>Pause time minus 28 slot times</td> </tr> <tr> <td>10</td> <td>Pause time minus 144 slot times</td> </tr> <tr> <td>11</td> <td>Pause time minus 256 slot times</td> </tr> </tbody> </table> <p>Slot time is defined as time taken to transmit 512 bits (64 bytes) on the GMII/MII interface.</p>	Selection	Threshold	00	Pause time minus 4 slot times	01	Pause time minus 28 slot times	10	Pause time minus 144 slot times	11	Pause time minus 256 slot times
Selection	Threshold												
00	Pause time minus 4 slot times												
01	Pause time minus 28 slot times												
10	Pause time minus 144 slot times												
11	Pause time minus 256 slot times												

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	UP Unicast Pause Frame Detect When this bit is set, the GMAC will detect the Pause frames with the station's unicast address specified in MAC Address0 High Register and MAC Address0 Low Register, in addition to the detecting Pause frames with the unique multicast address. When this bit is reset, the GMAC will detect only a Pause frame with the unique multicast address specified in the 802.3x standard.
2	RW	0x0	RFE Receive Flow Control Enable When this bit is set, the GMAC will decode the received Pause frame and disable its transmitter for a specified (Pause Time) time. When this bit is reset, the decode function of the Pause frame is disabled.
1	RW	0x0	TFE Transmit Flow Control Enable In Full-Duplex mode, when this bit is set, the GMAC enables the flow control operation to transmit Pause frames. When this bit is reset, the flow control operation in the GMAC is disabled, and the GMAC will not transmit any Pause frames. In Half-Duplex mode, when this bit is set, the GMAC enables the back-pressure operation. When this bit is reset, the backpressure feature is disabled.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	<p>FCB_BPA Flow Control Busy/Backpressure Activate This bit initiates a Pause Control frame in Full-Duplex mode and activates the backpressure function in Half-Duplex mode if TFE bit is set.</p> <p>In Full-Duplex mode, this bit should be read as 1'b0 before writing to the register GMAC_FLOW_CTRL. To initiate a pause control frame, the application must set this bit to 1'b1. During a transfer of the control frame, this bit will continue to be set to signify that a frame transmission is in progress. After the completion of Pause control frame transmission, the GMAC will reset this bit to 1'b0. The register GMAC_FLOW_CTRL should not be written to until this bit is cleared.</p> <p>In Half-Duplex mode, when this bit is set (and TFE is set), then backpressure is asserted by the GMAC Core. During backpressure, when the GMAC receives a new frame, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically OR'ed with the mti_flowctrl_i input signal for the backpressure function.</p>

**GMAC\_VLAN\_TAG**

Address: Operational Base + offset (0x001c)

VLAN Tag Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved
16	RW	0x0	<p>ETV Enable 12-Bit VLAN Tag Comparison When this bit is set, a 12-bit VLAN identifier, rather than the complete 16-bit VLAN tag, is used for comparison and filtering. Bits[11:0] of the VLAN tag are compared with the corresponding field in the received VLAN-tagged frame.</p> <p>When this bit is reset, all 16 bits of the received VLAN frame's fifteenth and sixteenth bytes are used for comparison.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	<p>VL VLAN Tag Identifier for Receive Frames This contains the 802.1Q VLAN tag to identify VLAN frames, and is compared to the fifteenth and sixteenth bytes of the frames being received for VLAN frames. Bits[15:13] are the User Priority, Bit[12] is the Canonical Format Indicator (CFI) and bits[11:0] are the VLAN tag's VLAN Identifier (VID) field. When the ETV bit is set, only the VID (Bits[11:0]) is used for comparison.</p> <p>If VL (VL[11:0] if ETV is set) is all zeros, the GMAC does not check the fifteenth and sixteenth bytes for VLAN tag comparison, and declares all frames with a Type field value of 0x8100 to be VLAN frames.</p>

**GMAC\_DEBUG**

Address: Operational Base + offset (0x0024)

Debug register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RO	0x0	reserved
25	RW	0x0	<p>TFIFO3 When high, it indicates that the MTL TxStatus FIFO is full and hence the MTL will not be accepting any more frames for transmission.</p>
24	RW	0x0	<p>TFIFO2 When high, it indicates that the MTL TxFIFO is not empty and has some data left for transmission.</p>
23	RO	0x0	reserved
22	RW	0x0	<p>TFIFO1 When high, it indicates that the MTL TxFIFO Write Controller is active and transferring data to the TxFIFO.</p>
21:20	RW	0x0	<p>TFIFOSTA This indicates the state of the TxFIFO read Controller: 2'b00: IDLE state 2'b01: READ state (transferring data to MAC transmitter) 2'b10: Waiting for TxStatus from MAC transmitter 2'b11: Writing the received TxStatus or flushing the TxFIFO</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19	RW	0x0	PAUSE When high, it indicates that the MAC transmitter is in PAUSE condition (in full-duplex only) and hence will not schedule any frame for transmission
18:17	RW	0x0	TSAT This indicates the state of the MAC Transmit Frame Controller module: 2'b00: IDLE 2'b01: Waiting for Status of previous frame or IFG/backoff period to be over 2'b10: Generating and transmitting a PAUSE control frame (in full duplex mode) 2'b11: Transferring input frame for transmission
16	RW	0x0	TACT When high, it indicates that the MAC GMII/MII transmit protocol engine is actively transmitting data and not in IDLE state.
15:10	RO	0x0	reserved
9:8	RW	0x0	RFIFO This gives the status of the RxFIFO Fill-level: 2'b00: RxFIFO Empty 2'b01: RxFIFO fill-level below flow-control de-activate threshold 2'b10: RxFIFO fill-level above flow-control activate threshold 2'b11: RxFIFO Full
7	RO	0x0	reserved
6:5	RW	0x0	RFIFORD It gives the state of the RxFIFO read Controller: 2'b00: IDLE state 2'b01: Reading frame data 2'b10: Reading frame status (or time-stamp) 2'b11: Flushing the frame data and Status
4	RW	0x0	RFIFOWR When high, it indicates that the MTL RxFIFO Write Controller is active and transferring a received frame to the FIFO.
3	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:1	RW	0x0	ACT When high, it indicates the active state of the small FIFO Read and Write controllers respectively of the MAC receive Frame Controller module
0	RW	0x0	RDB When high, it indicates that the MAC GMII/MII receive protocol engine is actively receiving data and not in IDLE state.

**GMAC\_PMT\_CTRL\_STA**

Address: Operational Base + offset (0x002c)

PMT Control and Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	W1C	0x0	WFFRPR Wake-Up Frame Filter Register Pointer Reset When set, resets the Remote Wake-up Frame Filter register pointer to 3'b000. It is automatically cleared after 1 clock cycle.
30:10	RO	0x0	reserved
9	RW	0x0	GU Global Unicast When set, enables any unicast packet filtered by the GMAC (DAF) address recognition to be a wake-up frame.
8:7	RO	0x0	reserved
6	RC	0x0	WFR Wake-Up Frame Received When set, this bit indicates the power management event was generated due to reception of a wake-up frame. This bit is cleared by a read into this register.
5	RC	0x0	MPR Magic Packet Received When set, this bit indicates the power management event was generated by the reception of a Magic Packet. This bit is cleared by a read into this register.
4:3	RO	0x0	reserved
2	RW	0x0	WFE Wake-Up Frame Enable When set, enables generation of a power management event due to wake-up frame reception.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	MPE Magic Packet Enable When set, enables generation of a power management event due to Magic Packet reception.
0	R/WSC	0x0	PD Power Down When set, all received frames will be dropped. This bit is cleared automatically when a magic packet or Wake-Up frame is received, and Power-Down mode is disabled. Frames received after this bit is cleared are forwarded to the application. This bit must only be set when either the Magic Packet Enable or Wake-Up Frame Enable bit is set high.

**GMAC\_INT\_STATUS**

Address: Operational Base + offset (0x0038)

Interrupt Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7	RO	0x0	MRCOIS MMC Receive Checksum Offload Interrupt Status This bit is set high whenever an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared.
6	RO	0x0	MTIS MMC Transmit Interrupt Status This bit is set high whenever an interrupt is generated in the MMC Transmit Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. This bit is only valid when the optional MMC module is selected during configuration.
5	RO	0x0	MRIS MMC Receive Interrupt Status This bit is set high whenever an interrupt is generated in the MMC Receive Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. This bit is only valid when the optional MMC module is selected during configuration.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RO	0x0	MIS MMC Interrupt Status This bit is set high whenever any of bits 7:5 is set high and cleared only when all of these bits are low. This bit is valid only when the optional MMC module is selected during configuration.
3	RO	0x0	PIS PMT Interrupt Status This bit is set whenever a Magic packet or Wake-on-LAN frame is received in Power-Down mode). This bit is cleared when both bits[6:5] are cleared due to a read operation to the register GMAC_PMT_CTRL_STA.
2:1	RO	0x0	reserved
0	RO	0x0	RIS RGMII Interrupt Status This bit is set due to any change in value of the Link Status of RGMII interface. This bit is cleared when the user makes a read operation to the RGMII Status register.

**GMAC\_INT\_MASK**

Address: Operational Base + offset (0x003c)

Interrupt Mask Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3	RW	0x0	PIM PMT Interrupt Mask This bit when set, will disable the assertion of the interrupt signal due to the setting of PMT Interrupt Status bit in Register GMAC_INT_STATUS.
2:1	RO	0x0	reserved
0	RW	0x0	RIM RGMII Interrupt Mask This bit when set, will disable the assertion of the interrupt signal due to the setting of RGMII Interrupt Status bit in Register GMAC_INT_STATUS.

**GMAC\_MAC\_ADDR0\_HI**

Address: Operational Base + offset (0x0040)

MAC Address0 High Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0xffff	A47_A32 MAC Address0 [47:32] This field contains the upper 16 bits (47:32) of the 6-byte first MAC address. This is used by the MAC for filtering for received frames and for inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

**GMAC\_MAC\_ADDR0\_LO**

Address: Operational Base + offset (0x0044)

MAC Address0 Low Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0xffffffff	A31_A0 MAC Address0 [31:0] This field contains the lower 32 bits of the 6-byte first MAC address. This is used by the MAC for filtering for received frames and for inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

**GMAC\_AN\_CTRL**

Address: Operational Base + offset (0x00c0)

AN Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12	RW	0x0	ANE Auto-Negotiation Enable When set, will enable the GMAC to perform auto-negotiation with the link partner. Clearing this bit will disable auto-negotiation.
11:10	RO	0x0	reserved
9	RWSC	0x0	RAN Restart Auto-Negotiation When set, will cause auto-negotiation to restart if the ANE is set. This bit is self-clearing after auto-negotiation starts. This bit should be cleared for normal operation.
8:0	RO	0x0	reserved

**GMAC\_AN\_STATUS**

Address: Operational Base + offset (0x00c4)

AN Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RO	0x0	ANC Auto-Negotiation Complete When set, this bit indicates that the auto-negotiation process is completed. This bit is cleared when auto-negotiation is reinitiated.
4	RO	0x0	reserved
3	RO	0x1	ANA Auto-Negotiation Ability This bit is always high, because the GMAC supports auto-negotiation.
2	RWSC	0x0	LS Link Status When set, this bit indicates that the link is up. When cleared, this bit indicates that the link is down.
1:0	RO	0x0	reserved

**GMAC\_AN\_ADV**

Address: Operational Base + offset (0x00c8)

Auto\_Negotiation Advertisement Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15	RO	0x0	NP Next Page Support This bit is tied to low, because the GMAC does not support the next page.
14	RO	0x0	reserved
13:12	RW	0x0	RFE Remote Fault Encoding These 2 bits provide a remote fault encoding, indicating to a link partner that a fault or error condition has occurred.
11:9	RO	0x0	reserved
8:7	RW	0x3	PSE Pause Encoding These 2 bits provide an encoding for the PAUSE bits, indicating that the GMAC is capable of configuring the PAUSE function as defined in IEEE 802.3x.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RW	0x1	HD Half-Duplex This bit, when set high, indicates that the GMAC supports Half-Duplex. This bit is tied to low (and RO) when the GMAC is configured for Full-Duplex-only operation.
5	RW	0x1	FD Full-Duplex This bit, when set high, indicates that the GMAC supports Full-Duplex.
4:0	RO	0x0	reserved

**GMAC\_AN\_LINK\_PART\_AB**

Address: Operational Base + offset (0x00cc)

Auto\_Negotiation Link Partner Ability Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15	RO	0x0	NP Next Page Support When set, this bit indicates that more next page information is available. When cleared, this bit indicates that next page exchange is not desired.
14	RO	0x0	ACK Acknowledge When set, this bit is used by the auto-negotiation function to indicate that the link partner has successfully received the GMAC's base page. When cleared, it indicates that a successful receipt of the base page has not been achieved.
13:12	RO	0x0	RFE Remote Fault Encoding These 2 bits provide a remote fault encoding, indicating a fault or error condition of the link partner.
11:9	RO	0x0	reserved
8:7	RO	0x0	PSE Pause Encoding These 2 bits provide an encoding for the PAUSE bits, indicating that the link partner's capability of configuring the PAUSE function as defined in IEEE 802.3x.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
6	RO	0x0	HD Half-Duplex When set, this bit indicates that the link partner has the ability to operate in Half-Duplex mode. When cleared, the link partner does not have the ability to operate in Half-Duplex mode.
5	RO	0x0	FD Full-Duplex When set, this bit indicates that the link partner has the ability to operate in Full-Duplex mode. When cleared, the link partner does not have the ability to operate in Full-Duplex mode.
4:0	RO	0x0	reserved

**GMAC\_AN\_EXP**

Address: Operational Base + offset (0x00d0)

Auto\_Negotiation Expansion Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2	RO	0x0	NPA Next Page Ability This bit is tied to low, because the GMAC does not support next page function.
1	RO	0x0	NPR New Page Received When set, this bit indicates that a new page has been received by the GMAC. This bit will be cleared when read.
0	RO	0x0	reserved

**GMAC\_INTF\_MODE\_STA**

Address: Operational Base + offset (0x00d8)

RGMII Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3	RO	0x0	LST Link Status Indicates whether the link is up (1'b1) or down (1'b0)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2:1	RO	0x0	LSD Link Speed Indicates the current speed of the link: 2'b00: 2.5 MHz 2'b01: 25 MHz 2'b10: 125 MHz
0	RW	0x0	LM Link Mode Indicates the current mode of operation of the link: 1'b0: Half-Duplex mode 1'b1: Full-Duplex mode

**GMAC\_MMCTRL**

Address: Operational Base + offset (0x0100)

MMC Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5	RW	0x0	FHP Full-Half preset When low and bit4 is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0xFFFF_F800 (half - 2KBytes) and all frame-counters gets preset to 0xFFFF_FFF0 (half - 16) When high and bit4 is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (full - 2KBytes) and all frame-counters gets preset to 0xFFFF_FFF0 (full - 16)
4	RWSC	0x0	CP Counters Preset When set, all counters will be initialized or preset to almost full or almost half as per Bit5 above. This bit will be cleared automatically after 1 clock cycle. This bit along with bit5 is useful for debugging and testing the assertion of interrupts due to MMC counter becoming half-full or full.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	MCF MMC Counter Freeze When set, this bit freezes all the MMC counters to their current value. (None of the MMC counters are updated due to any transmitted or received frame until this bit is reset to 0. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.)
2	RW	0x0	ROR Reset on Read When set, the MMC counters will be reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (bits[7:0]) is read.
1	RW	0x0	CSR Counter Stop Rollover When set, counter after reaching maximum value will not roll over to zero
0	RWSC	0x0	CR Counters Reset When set, all counters will be reset. This bit will be cleared automatically after 1 clock cycle

**GMAC\_MMCR\_RX\_INTR**

Address: Operational Base + offset (0x0104)

MMC Receive Interrupt Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RO	0x0	reserved
21	RW	0x0	INT21 The bit is set when the rxfifooverflow counter reaches half the maximum value, and also when it reaches the maximum value.
20:19	RO	0x0	reserved
18	RC	0x0	INT18 The bit is set when the rxlengtherror counter reaches half the maximum value, and also when it reaches the maximum value.
17:6	RO	0x0	reserved
5	RW	0x0	INT5 The bit is set when the rxcrcerror counter reaches half the maximum value, and also when it reaches the maximum value.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RC	0x0	INT4 The bit is set when the rxmulticastframes_g counter reaches half the maximum value, and also when it reaches the maximum value.
3	RO	0x0	reserved
2	RC	0x0	INT2 The bit is set when the rxoctetcount_g counter reaches half the maximum value, and also when it reaches the maximum value.
1	RC	0x0	INT1 The bit is set when the rxoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value.
0	RC	0x0	INT0 The bit is set when the rxframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

**GMAC\_MMC\_TX\_INTR**

Address: Operational Base + offset (0x0108)

MMC Transmit Interrupt Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RO	0x0	reserved
21	RC	0x0	INT21 The bit is set when the txframecount_g counter reaches half the maximum value, and also when it reaches the maximum value.
20	RC	0x0	INT20 The bit is set when the txoctetcount_g counter reaches half the maximum value, and also when it reaches the maximum value.
19	RC	0x0	INT19 The bit is set when the txcarriererror counter reaches half the maximum value, and also when it reaches the maximum value.
18:14	RO	0x0	reserved
13	RC	0x0	INT13 The bit is set when the txunderflowerror counter reaches half the maximum value, and also when it reaches the maximum value.
12:2	RO	0x0	reserved
1	RC	0x0	INT1 The bit is set when the txframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RC	0x0	INT0 The bit is set when the txoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

**GMAC\_MMC\_RX\_INT\_MSK**

Address: Operational Base + offset (0x010c)

MMC Receive Interrupt Mask Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RO	0x0	reserved
21	RW	0x0	INT21 Setting this bit masks the interrupt when the rx fifo overflow counter reaches half the maximum value, and also when it reaches the maximum value.
20:19	RO	0x0	reserved
18	RW	0x0	INT18 Setting this bit masks the interrupt when the rx length error counter reaches half the maximum value, and also when it reaches the maximum value.
17:6	RO	0x0	reserved
5	RW	0x0	INT5 Setting this bit masks the interrupt when the rx crc error counter reaches half the maximum value, and also when it reaches the maximum value.
4	RW	0x0	INT4 Setting this bit masks the interrupt when the rx multicast frames_g counter reaches half the maximum value, and also when it reaches the maximum value.
3	RO	0x0	reserved
2	RW	0x0	INT2 Setting this bit masks the interrupt when the rx octet count_g counter reaches half the maximum value, and also when it reaches the maximum value.
1	RW	0x0	INT1 Setting this bit masks the interrupt when the rx octet count_gb counter reaches half the maximum value, and also when it reaches the maximum value.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	INT0 Setting this bit masks the interrupt when the rxframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

**GMAC\_MMC\_TX\_INT\_MSK**

Address: Operational Base + offset (0x0110)

MMC Transmit Interrupt Mask Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:22	RO	0x0	reserved
21	RW	0x0	INT21 Setting this bit masks the interrupt when the txframecount_g counter reaches half the maximum value, and also when it reaches the maximum value.
20	RW	0x0	INT20 Setting this bit masks the interrupt when the txoctetcount_g counter reaches half the maximum value, and also when it reaches the maximum value.
19	RW	0x0	INT19 Setting this bit masks the interrupt when the txcarriererror counter reaches half the maximum value, and also when it reaches the maximum value.
18:14	RO	0x0	reserved
13	RW	0x0	INT13 Setting this bit masks the interrupt when the txunderflowerror counter reaches half the maximum value, and also when it reaches the maximum value.
12:2	RO	0x0	reserved
1	RW	0x0	INT1 Setting this bit masks the interrupt when the txframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.
0	RW	0x0	INT0 Setting this bit masks the interrupt when the txoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

**GMAC\_MMC\_TXOCTETCNT\_GB**

Address: Operational Base + offset (0x0114)

MMC TX OCTET Good and Bad Counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txoctetcount_gb Number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad frames.

### **GMAC\_MMC\_TXFRMCNT\_GB**

Address: Operational Base + offset (0x0118)

MMC TX Frame Good and Bad Counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txframecount_gb Number of good and bad frames transmitted, exclusive of retried frames.

### **GMAC\_MMC\_TXUNDFLWERR**

Address: Operational Base + offset (0x0148)

MMC TX Underflow Error

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txunderflowerror Number of frames aborted due to frame underflow error.

### **GMAC\_MMC\_TXCARERR**

Address: Operational Base + offset (0x0160)

MMC TX Carrier Error

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txcarriererror Number of frames aborted due to carrier sense error (no carrier or loss of carrier).

### **GMAC\_MMC\_TXOCTETCNT\_G**

Address: Operational Base + offset (0x0164)

MMC TX OCTET Good Counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txoctetcount_g Number of bytes transmitted, exclusive of preamble, in good frames only.

### **GMAC\_MMC\_TXFRMCNT\_G**

Address: Operational Base + offset (0x0168)

MMC TX Frame Good Counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txframecount_g Number of good frames transmitted.

**GMAC\_MMC\_RXFRMCNT\_GB**

Address: Operational Base + offset (0x0180)

MMC RX Frame Good and Bad Counter

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	rxframecount_gb Number of good and bad frames received.

**GMAC\_MMC\_RXOCTETCNT\_GB**

Address: Operational Base + offset (0x0184)

MMC RX OCTET Good and Bad Counter

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	rxoctetcount_gb Number of bytes received, exclusive of preamble, in good and bad frames.

**GMAC\_MMC\_RXOCTETCNT\_G**

Address: Operational Base + offset (0x0188)

MMC RX OCTET Good Counter

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	rxoctetcount_g Number of bytes received, exclusive of preamble, only in good frames.

**GMAC\_MMC\_RXMCFRMCNT\_G**

Address: Operational Base + offset (0x0190)

MMC RX Multicast Frame Good Counter

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	rxmulticastframes_g Number of good multicast frames received.

**GMAC\_MMC\_RXCRCERR**

Address: Operational Base + offset (0x0194)

MMC RX Carrier

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	rxcrcerror Number of frames received with CRC error.

**GMAC\_MMC\_RXLENERR**

Address: Operational Base + offset (0x01c8)

MMC RX Length Error

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	rxlengtherror Number of frames received with length error (Length type field ≠ frame size), for all frames with valid length field.

**GMAC\_MMC\_RXFIFOVRFLW**

Address: Operational Base + offset (0x01d4)

MMC RX FIFO Overflow

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxfifooverflow Number of missed received frames due to FIFO overflow.

**GMAC\_MMC\_IPC\_INT\_MSK**

Address: Operational Base + offset (0x0200)

MMC Receive Checksum Offload Interrupt Mask Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved
29	RW	0x0	INT29 Setting this bit masks the interrupt when the rxicmp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
28	RO	0x0	reserved
27	RW	0x0	INT27 Setting this bit masks the interrupt when the rxtcp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
26	RO	0x0	reserved
25	RW	0x0	INT25 Setting this bit masks the interrupt when the rxudp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
24:23	RO	0x0	reserved
22	RW	0x0	INT22 Setting this bit masks the interrupt when the rxipv6_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.
21:18	RO	0x0	reserved
17	RW	0x0	INT17 Setting this bit masks the interrupt when the rxipv4_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.
16:14	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13	RW	0x0	INT13 Setting this bit masks the interrupt when the rxicmp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
12	RO	0x0	reserved
11	RW	0x0	INT11 Setting this bit masks the interrupt when the rxtcp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
10	RO	0x0	reserved
9	RW	0x0	INT9 Setting this bit masks the interrupt when the rxudp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
8:7	RO	0x0	reserved
6	RW	0x0	INT6 Setting this bit masks the interrupt when the rxipv6_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.
5	RW	0x0	INT5 Setting this bit masks the interrupt when the rxipv6_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
4:2	RO	0x0	reserved
1	RW	0x0	INT1 Setting this bit masks the interrupt when the rxipv4_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.
0	RW	0x0	INT0 Setting this bit masks the interrupt when the rxipv4_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.

**GMAC\_MMIC\_IPC\_INTR**

Address: Operational Base + offset (0x0208)

MMC Receive Checksum Offload Interrupt Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:30	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
29	RC	0x0	INT29 The bit is set when the rxicmp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
28	RO	0x0	reserved
27	RC	0x0	INT27 The bit is set when the rxtcp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
26	RO	0x0	reserved
25	RC	0x0	INT25 The bit is set when the rxudp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
24:23	RO	0x0	reserved
22	RC	0x0	INT22 The bit is set when the rxipv6_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.
21:18	RO	0x0	reserved
17	RC	0x0	INT17 The bit is set when the rxipv4_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.
16:14	RO	0x0	reserved
13	RC	0x0	INT13 The bit is set when the rxicmp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
12	RO	0x0	reserved
11	RC	0x0	INT11 The bit is set when the rxtcp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
10	RO	0x0	reserved
9	RC	0x0	INT9 The bit is set when the rxudp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
8:7	RO	0x0	reserved
6	RC	0x0	INT6 The bit is set when the rxipv6_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RC	0x0	INT5 The bit is set when the rxipv6_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
4:2	RO	0x0	reserved
1	RC	0x0	INT1 The bit is set when the rxipv4_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.
0	RC	0x0	INT0 The bit is set when the rxipv4_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.

**GMAC\_MMC\_RXIPV4GFRM**

Address: Operational Base + offset (0x0210)

MMC RX IPV4 Good Frame

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxipv4_gd_frms Number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload

**GMAC\_MMC\_RXIPV4HDERRFRM**

Address: Operational Base + offset (0x0214)

MMC RX IPV4 Head Error Frame

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxipv4_hdrerr_frms Number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors

**GMAC\_MMC\_RXIPV6GFRM**

Address: Operational Base + offset (0x0224)

MMC RX IPV6 Good Frame

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxipv6_gd_frms Number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads.

**GMAC\_MMC\_RXIPV6HDERRFRM**

Address: Operational Base + offset (0x0228)

MMC RX IPV6 Head Error Frame

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxipv6_hdrerr_frms Number of IPv6 datagrams received with header errors (length or version mismatch).

**GMAC\_MMC\_RXUDPERRFRM**

Address: Operational Base + offset (0x0234)

MMC RX UDP Error Frame

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxudp_err_frms Number of good IP datagrams whose UDP payload has a checksum error.

**GMAC\_MMC\_RXTCPERRFRM**

Address: Operational Base + offset (0x023c)

MMC RX TCP Error Frame

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxtcp_err_frms Number of good IP datagrams whose TCP payload has a checksum error.

**GMAC\_MMC\_RXICMPERRFRM**

Address: Operational Base + offset (0x0244)

MMC RX ICMP Error Frame

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxicmp_err_frms Number of good IP datagrams whose ICMP payload has a checksum error.

**GMAC\_MMC\_RXIPV4HDRROCT**

Address: Operational Base + offset (0x0254)

MMC RX OCTET IPV4 Head Error

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxipv4_hdrerr_octets Number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter.

**GMAC\_MMC\_RXIPV6HDRROCT**

Address: Operational Base + offset (0x0268)

MMC RX OCTET IPV6 Head Error

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxipv6_hdrerr_octets Number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the IPv6 header's Length field is used to update this counter.

**GMAC\_MMC\_RXUDPERROCT**

Address: Operational Base + offset (0x0274)

MMC RX OCTET UDP Error

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxudp_err_octets Number of bytes received in a UDP segment that had checksum errors.

**GMAC\_MMC\_RXTCPPERROCT**

Address: Operational Base + offset (0x027c)

MMC RX OCTET TCP Error

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxtcp_err_octets Number of bytes received in a TCP segment with checksum errors.

**GMAC\_MMC\_RXICMPERROCT**

Address: Operational Base + offset (0x0284)

MMC RX OCTET ICMP Error

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	rxicmp_err_octets Number of bytes received in an ICMP segment with checksum errors.

**GMAC\_BUS\_MODE**

Address: Operational Base + offset (0x1000)

Bus Mode Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:26	RO	0x0	reserved
25	RW	0x0	AAL Address-Aligned Beats When this bit is set high and the FB bit equals 1, the AXI interface generates all bursts aligned to the start address LS bits. If the FB bit equals 0, the first burst (accessing the data buffer's start address) is not aligned, but subsequent bursts are aligned to the address.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
24	RW	0x0	PBL_Mode 8xPBL Mode When set high, this bit multiplies the PBL value programmed (bits [22:17] and bits [13:8]) eight times. Thus the DMA will transfer data in to a maximum of 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.
23	RW	0x0	USP Use Separate PBL When set high, it configures the RxDMA to use the value configured in bits [22:17] as PBL while the PBL value in bits [13:8] is applicable to TxDMA operations only. When reset to low, the PBL value in bits [13:8] is applicable for both DMA engines.
22:17	RW	0x01	RPBL RxDMA PBL These bits indicate the maximum number of beats to be transferred in one RxDMA transaction. This will be the maximum value that is used in a single block Read/Write. The RxDMA will always attempt to burst as specified in RPBL each time it starts a Burst transfer on the host bus. RPBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value will result in undefined behavior. These bits are valid and applicable only when USP is set high.
16	RW	0x0	FB Fixed Burst This bit controls whether the AXI Master interface performs fixed burst transfers or not. When set, the AHB will use only SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers. When reset, the AXI will use SINGLE and INCR burst transfer operations.
15:14	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13:8	RW	0x01	<p>PBL Programmable Burst Length These bits indicate the maximum number of beats to be transferred in one DMA transaction. This will be the maximum value that is used in a single block Read/Write. The DMA will always attempt to burst as specified in PBL each time it starts a Burst transfer on the host bus. PBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value will result in undefined behavior. When USP is set high, this PBL value is applicable for TxDMA transactions only.</p> <p>The PBL values have the following limitations. The maximum number of beats (PBL) possible is limited by the size of the Tx FIFO and Rx FIFO in the MTL layer and the data bus width on the DMA. The FIFO has a constraint that the maximum beat supported is half the depth of the FIFO, except when specified (as given below). For different data bus widths and FIFO sizes, the valid PBL range (including x8 mode) is provided in the following table. If the PBL is common for both transmit and receive DMA, the minimum Rx FIFO and Tx FIFO depths must be considered. Do not program out-of-range PBL values, because the system may not behave properly.</p> <p>For TxFIFO, valid PBL range in full duplex mode and duplex mode is 128 or less.</p> <p>For RxFIFO, valid PBL range in full duplex mode is all.</p>
7	RO	0x0	reserved
6:2	RW	0x00	<p>DSL Descriptor Skip Length This bit specifies the number of Dword to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor. When DSL value equals zero, then the descriptor table is taken as contiguous by the DMA, in Ring mode.</p>
1	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	R/WSC	0x1	<p>SWR Software Reset</p> <p>When this bit is set, the MAC DMA Controller resets all GMAC Subsystem internal registers and logic. It is cleared automatically after the reset operation has completed in all of the core clock domains. Read a 0 value in this bit before re-programming any register of the core.</p> <p>Note: The reset operation is completed only when all the resets in all the active clock domains are de-asserted. Hence it is essential that all the PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion.</p>

**GMAC\_TX\_POLL\_DEMAND**

Address: Operational Base + offset (0x1004)

Transmit Poll Demand Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	<p>TPD Transmit Poll Demand</p> <p>When these bits are written with any value, the DMA reads the current descriptor pointed to by Register GMAC_CUR_HOST_TX_DESC. If that descriptor is not available (owned by Host), transmission returns to the Suspend state and DMA Register GMAC_STATUS[2] is asserted. If the descriptor is available, transmission resumes.</p>

**GMAC\_RX\_POLL\_DEMAND**

Address: Operational Base + offset (0x1008)

Receive Poll Demand Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	<p>RPD Receive Poll Demand</p> <p>When these bits are written with any value, the DMA reads the current descriptor pointed to by Register GMAC_CUR_HOST_RX_DESC. If that descriptor is not available (owned by Host), reception returns to the Suspended state and Register GMAC_STATUS[7] is not asserted. If the descriptor is available, the Receive DMA returns to active state.</p>

**GMAC\_RX\_DESC\_LIST\_ADDR**

Address: Operational Base + offset (0x100c)

Receive Descriptor List Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	SRL Start of Receive List This field contains the base address of the First Descriptor in the Receive Descriptor list. The LSB bits [1/2/3:0] for 32/64/128-bit bus width) will be ignored and taken as all-zero by the DMA internally. Hence these LSB bits are Read Only.

**GMAC\_TX\_DESC\_LIST\_ADDR**

Address: Operational Base + offset (0x1010)

Transmit Descriptor List Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	STL Start of Transmit List This field contains the base address of the First Descriptor in the Transmit Descriptor list. The LSB bits [1/2/3:0] for 32/64/128-bit bus width) will be ignored and taken as all-zero by the DMA internally. Hence these LSB bits are Read Only.

**GMAC\_STATUS**

Address: Operational Base + offset (0x1014)

Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28	RO	0x0	GPI GMAC PMT Interrupt This bit indicates an interrupt event in the GMAC core's PMT module. The software must read the corresponding registers in the GMAC core to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. The interrupt signal from the GMAC subsystem (sbd_intr_o) is high when this bit is high.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27	RO	0x0	GMI GMAC MMC Interrupt This bit reflects an interrupt event in the MMC module of the GMAC core. The software must read the corresponding registers in the GMAC core to get the exact cause of interrupt and clear the source of interrupt to make this bit as 1'b0. The interrupt signal from the GMAC subsystem (sbd_intr_o) is high when this bit is high.
26	RO	0x0	GLI GMAC Line interface Interrupt This bit reflects an interrupt event in the GMAC Core's PCS or RGMII interface block. The software must read the corresponding registers in the GMAC core to get the exact cause of interrupt and clear the source of interrupt to make this bit as 1'b0. The interrupt signal from the GMAC subsystem (sbd_intr_o) is high when this bit is high.
25:23	RO	0x0	EB Error Bits These bits indicate the type of error that caused a Bus Error (e.g., error response on the AXI interface). Valid only with Fatal Bus Error bit (Register GMAC_STATUS[13]) set. This field does not generate an interrupt. Bit 23: 1'b1 Error during data transfer by TxDMA 1'b0 Error during data transfer by RxDMA Bit 24: 1'b1 Error during read transfer 1'b0 Error during write transfer Bit 25: 1'b1 Error during descriptor access 1'b0 Error during data buffer access

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
22:20	RO	0x0	<p>TS Transmit Process State These bits indicate the Transmit DMA FSM state. This field does not generate an interrupt.</p> <p>3'b000: Stopped; Reset or Stop Transmit Command issued.</p> <p>3'b001: Running; Fetching Transmit Transfer Descriptor.</p> <p>3'b010: Running; Waiting for status.</p> <p>3'b011: Running; Reading Data from host memory buffer and queuing it to transmit buffer (Tx FIFO).</p> <p>3'b100: TIME_STAMP write state.</p> <p>3'b101: Reserved for future use.</p> <p>3'b110: Suspended; Transmit Descriptor Unavailable or Transmit Buffer Underflow.</p> <p>3'b111: Running; Closing Transmit Descriptor.</p>
19:17	RO	0x0	<p>RS Receive Process State These bits indicate the Receive DMA FSM state. This field does not generate an interrupt.</p> <p>3'b000: Stopped: Reset or Stop Receive Command issued.</p> <p>3'b001: Running: Fetching Receive Transfer Descriptor.</p> <p>3'b010: Reserved for future use.</p> <p>3'b011: Running: Waiting for receive packet.</p> <p>3'b100: Suspended: Receive Descriptor Unavailable.</p> <p>3'b101: Running: Closing Receive Descriptor.</p> <p>3'b110: TIME_STAMP write state.</p> <p>3'b111: Running: Transferring the receive packet data from receive buffer to host memory.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16	W1C	0x0	<p>NIS Normal Interrupt Summary Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in Register OP_MODE:</p> <ul style="list-style-type: none"> <li>Register GMAC_STATUS[0]: Transmit Interrupt</li> <li>Register GMAC_STATUS[2]: Transmit Buffer Unavailable</li> <li>Register GMAC_STATUS[6]: Receive Interrupt</li> <li>Register GMAC_STATUS[14]: Early Receive Interrupt</li> </ul> <p>Only unmasked bits affect the Normal Interrupt Summary bit. This is a sticky bit and must be cleared (by writing a 1 to this bit) each time a corresponding bit that causes NIS to be set is cleared.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	W1C	0x0	<p>AIS Abnormal Interrupt Summary Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in Register OP_MODE:</p> <ul style="list-style-type: none"> <li>Register GMAC_STATUS[1]: Transmit Process Stopped</li> <li>Register GMAC_STATUS[3]: Transmit Jabber Timeout</li> <li>Register GMAC_STATUS[4]: Receive FIFO Overflow</li> <li>Register GMAC_STATUS[5]: Transmit Underflow</li> <li>Register GMAC_STATUS[7]: Receive Buffer Unavailable</li> <li>Register GMAC_STATUS[8]: Receive Process Stopped</li> <li>Register GMAC_STATUS[9]: Receive Watchdog Timeout</li> <li>Register GMAC_STATUS[10]: Early Transmit Interrupt</li> <li>Register GMAC_STATUS[13]: Fatal Bus Error</li> </ul> <p>Only unmasked bits affect the Abnormal Interrupt Summary bit.</p> <p>This is a sticky bit and must be cleared each time a corresponding bit that causes AIS to be set is cleared.</p>
14	W1C	0x0	<p>ERI Early Receive Interrupt</p> <p>This bit indicates that the DMA had filled the first data buffer of the packet. Receive Interrupt Register GMAC_STATUS[6] automatically clears this bit.</p>
13	W1C	0x0	<p>FBI Fatal Bus Error Interrupt</p> <p>This bit indicates that a bus error occurred, as detailed in [25:23]. When this bit is set, the corresponding DMA engine disables all its bus accesses.</p>
12:11	RO	0x0	reserved
10	W1C	0x0	<p>ETI Early Transmit Interrupt</p> <p>This bit indicates that the frame to be transmitted was fully transferred to the MTL Transmit FIFO.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	W1C	0x0	RWT Receive Watchdog Timeout This bit is asserted when a frame with a length greater than 2,048 bytes is received.
8	W1C	0x0	RPS Receive Process Stopped This bit is asserted when the Receive Process enters the Stopped state.
7	W1C	0x0	RU Receive Buffer Unavailable This bit indicates that the Next Descriptor in the Receive List is owned by the host and cannot be acquired by the DMA. Receive Process is suspended. To resume processing Receive descriptors, the host should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, Receive Process resumes when the next recognized incoming frame is received. Register GMAC_STATUS[7] is set only when the previous Receive Descriptor was owned by the DMA.
6	W1C	0x0	RI Receive Interrupt This bit indicates the completion of frame reception. Specific frame status information has been posted in the descriptor. Reception remains in the Running state.
5	W1C	0x0	UNF Transmit Underflow This bit indicates that the Transmit Buffer had an Underflow during frame transmission. Transmission is suspended and an Underflow Error TDES0[1] is set.
4	W1C	0x0	OVF Receive Overflow This bit indicates that the Receive Buffer had an Overflow during frame reception. If the partial frame is transferred to application, the overflow status is set in RDES0[11].

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	W1C	0x0	TJT Transmit Jabber Timeout This bit indicates that the Transmit Jabber Timer expired, meaning that the transmitter had been excessively active. The transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Timeout TDES0[14] flag to assert.
2	W1C	0x0	TU Transmit Buffer Unavailable This bit indicates that the Next Descriptor in the Transmit List is owned by the host and cannot be acquired by the DMA. Transmission is suspended. Bits[22:20] explain the Transmit Process state transitions. To resume processing transmit descriptors, the host should change the ownership of the bit of the descriptor and then issue a Transmit Poll Demand command.
1	W1C	0x0	TPS Transmit Process Stopped This bit is set when the transmission is stopped.
0	W1C	0x0	TI Transmit Interrupt This bit indicates that frame transmission is finished and TDES1[31] is set in the First Descriptor.

**GMAC\_OP\_MODE**

Address: Operational Base + offset (0x1018)

Operation Mode Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:27	RO	0x0	reserved
26	RW	0x0	DT Disable Dropping of TCP/IP Checksum Error Frames When this bit is set, the core does not drop frames that only have errors detected by the Receive Checksum Offload engine. Such frames do not have any errors (including FCS error) in the Ethernet frame received by the MAC but have errors in the encapsulated payload only. When this bit is reset, all error frames are dropped if the FEF bit is reset.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
25	RW	0x0	RSF Receive Store and Forward When this bit is set, the MTL only reads a frame from the Rx FIFO after the complete frame has been written to it, ignoring RTC bits. When this bit is reset, the Rx FIFO operates in Cut-Through mode, subject to the threshold specified by the RTC bits.
24	RW	0x0	DFF Disable Flushing of Received Frames When this bit is set, the RxDMA does not flush any frames due to the unavailability of receive descriptors/buffers as it does normally when this bit is reset.
23:22	RO	0x0	reserved
21	RW	0x0	TSF Transmit Store and Forward When this bit is set, transmission starts when a full frame resides in the MTL Transmit FIFO. When this bit is set, the TTC values specified in Register GMAC_OP_MODE[16:14] are ignored. This bit should be changed only when transmission is stopped.
20	W1C	0x0	FTF Flush Transmit FIFO When this bit is set, the transmit FIFO controller logic is reset to its default values and thus all data in the Tx FIFO is lost/flushed. This bit is cleared internally when the flushing operation is completed fully. The Operation Mode register should not be written to until this bit is cleared. The data which is already accepted by the MAC transmitter will not be flushed. It will be scheduled for transmission and will result in underflow and runt frame transmission.  Note: The flush operation completes only after emptying the Tx FIFO of its contents and all the pending Transmit Status of the transmitted frames are accepted by the host. In order to complete this flush operation, the PHY transmit clock (clk_tx_i) is required to be active.
19:17	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>																
16:14	RW	0x0	<p>TTC Transmit Threshold Control These three bits control the threshold level of the MTL Transmit FIFO. Transmission starts when the frame size within the MTL Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when the TSF bit (Bit 21) is reset.</p> <table> <tr><td>3'b000:</td><td>64</td></tr> <tr><td>3'b001:</td><td>128</td></tr> <tr><td>3'b010:</td><td>192</td></tr> <tr><td>3'b011:</td><td>256</td></tr> <tr><td>3'b100:</td><td>40</td></tr> <tr><td>3'b101:</td><td>32</td></tr> <tr><td>3'b110:</td><td>24</td></tr> <tr><td>3'b111:</td><td>16</td></tr> </table>	3'b000:	64	3'b001:	128	3'b010:	192	3'b011:	256	3'b100:	40	3'b101:	32	3'b110:	24	3'b111:	16
3'b000:	64																		
3'b001:	128																		
3'b010:	192																		
3'b011:	256																		
3'b100:	40																		
3'b101:	32																		
3'b110:	24																		
3'b111:	16																		

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13	RW	0x0	<p>ST Start/Stop Transmission Command When this bit is set, transmission is placed in the Running state, and the DMA checks the Transmit List at the current position for a frame to be transmitted. Descriptor acquisition is attempted either from the current position in the list, which is the Transmit List Base Address set by Register GMAC_TX_DESC_LIST_ADDR, or from the position retained when transmission was stopped previously. If the current descriptor is not owned by the DMA, transmission enters the Suspended state and Transmit Buffer Unavailable (Register GMAC_STATUS[2]) is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting DMA Register TX_DESC_LIST_ADDR, then the DMA behavior is unpredictable.</p> <p>When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and becomes the current position when transmission is restarted. The stop transmission command is effective only the transmission of the current frame is complete or when the transmission is in the Suspended state.</p>
12:11	RW	0x0	<p>RFD Threshold for deactivating flow control (in both HD and FD) These bits control the threshold (Fill-level of Rx FIFO) at which the flow-control is deasserted after activation.</p> <p>2'b00: Full minus 1 KB 2'b01: Full minus 2 KB 2'b10: Full minus 3 KB 2'b11: Full minus 4 KB Note that the deassertion is effective only after flow control is asserted.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
10:9	RW	0x0	<p>RFA</p> <p>Threshold for activating flow control (in both HD and FD)</p> <p>These bits control the threshold (Fill level of Rx FIFO) at which flow control is activated.</p> <p>2'b00: Full minus 1 KB 2'b01: Full minus 2 KB 2'b10: Full minus 3 KB 2'b11: Full minus 4 KB</p> <p>Note that the above only applies to Rx FIFOs of 4 KB or more when the EFC bit is set high.</p>
8	RW	0x0	<p>EFC</p> <p>Enable HW flow control</p> <p>When this bit is set, the flow control signal operation based on fill-level of Rx FIFO is enabled. When reset, the flow control operation is disabled.</p>
7	RW	0x0	<p>FEF</p> <p>Forward Error Frames</p> <p>When this bit is reset, the Rx FIFO drops frames with error status (CRC error, collision error, GMII_ER, giant frame, watchdog timeout, overflow). However, if the frame's start byte (write) pointer is already transferred to the read controller side (in Threshold mode), then the frames are not dropped.</p> <p>When FEF is set, all frames except runt error frames are forwarded to the DMA. But when RxFIFO overflows when a partial frame is written, then such frames are dropped even when FEF is set.</p>
6	RW	0x0	<p>FUF</p> <p>Forward Undersized Good Frames</p> <p>When set, the Rx FIFO will forward Undersized frames (frames with no Error and length less than 64 bytes) including pad-bytes and CRC). When reset, the Rx FIFO will drop all frames of less than 64 bytes, unless it is already transferred due to lower value of Receive Threshold (e.g., RTC = 01).</p>
5	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:3	RW	0x0	<p>RTC</p> <p>Receive Threshold Control</p> <p>These two bits control the threshold level of the MTL Receive FIFO. Transfer (request) to DMA starts when the frame size within the MTL Receive FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are transferred automatically.</p> <p>Note that value of 11 is not applicable if the configured Receive FIFO size is 128 bytes.</p> <p>These bits are valid only when the RSF bit is zero, and are ignored when the RSF bit is set to 1.</p> <p>2'b00: 64 2'b01: 32 2'b10: 96 2'b11: 128</p>
2	RW	0x0	<p>OSF</p> <p>Operate on Second Frame</p> <p>When this bit is set, this bit instructs the DMA to process a second frame of Transmit data even before status for first frame is obtained.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	<p>SR Start/Stop Receive When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes incoming frames. Descriptor acquisition is attempted from the current position in the list, which is the address set by register GMAC_RX_DESC_LIST_ADDR or the position retained when the Receive process was previously stopped. If no descriptor is owned by the DMA, reception is suspended and Receive Buffer Unavailable (Register GMAC_STATUS[7]) is set. The Start Receive command is effective only when reception has stopped. If the command was issued before setting register GMAC_RX_DESC_LIST_ADDR, DMA behavior is unpredictable. When this bit is cleared, RxDMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state.</p>
0	RO	0x0	reserved

**GMAC\_INT\_ENA**

Address: Operational Base + offset (0x101c)

Interrupt Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:17	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
16	RW	0x0	<p>NIE Normal Interrupt Summary Enable When this bit is set, a normal interrupt is enabled. When this bit is reset, a normal interrupt is disabled. This bit enables the following bits:</p> <ul style="list-style-type: none"> <li>Register GMAC_STATUS[0]: Transmit Interrupt</li> <li>Register GMAC_STATUS[2]: Transmit Buffer Unavailable</li> <li>Register GMAC_STATUS[6]: Receive Interrupt</li> <li>Register GMAC_STATUS[14]: Early Receive Interrupt</li> </ul>
15	RW	0x0	<p>AIE Abnormal Interrupt Summary Enable When this bit is set, an Abnormal Interrupt is enabled. When this bit is reset, an Abnormal Interrupt is disabled. This bit enables the following bits</p> <ul style="list-style-type: none"> <li>Register GMAC_STATUS[1]: Transmit Process Stopped</li> <li>Register GMAC_STATUS[3]: Transmit Jabber Timeout</li> <li>Register GMAC_STATUS[4]: Receive Overflow</li> <li>Register GMAC_STATUS[5]: Transmit Underflow</li> <li>Register GMAC_STATUS[7]: Receive Buffer Unavailable</li> <li>Register GMAC_STATUS[8]: Receive Process Stopped</li> <li>Register GMAC_STATUS[9]: Receive Watchdog Timeout</li> <li>Register GMAC_STATUS[10]: Early Transmit Interrupt</li> <li>Register GMAC_STATUS[13]: Fatal Bus Error</li> </ul>
14	RW	0x0	<p>ERE Early Receive Interrupt Enable When this bit is set with Normal Interrupt Summary Enable (BIT 16), Early Receive Interrupt is enabled. When this bit is reset, Early Receive Interrupt is disabled.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13	RW	0x0	FBE Fatal Bus Error Enable When this bit is set with Abnormal Interrupt Summary Enable (BIT 15), the Fatal Bus Error Interrupt is enabled. When this bit is reset, Fatal Bus Error Enable Interrupt is disabled.
12:11	RO	0x0	reserved
10	RW	0x0	ETE Early Transmit Interrupt Enable When this bit is set with an Abnormal Interrupt Summary Enable (BIT 15), Early Transmit Interrupt is enabled. When this bit is reset, Early Transmit Interrupt is disabled.
9	RW	0x0	RWE Receive Watchdog Timeout Enable When this bit is set with Abnormal Interrupt Summary Enable (BIT 15), the Receive Watchdog Timeout Interrupt is enabled. When this bit is reset, Receive Watchdog Timeout Interrupt is disabled.
8	RW	0x0	RSE Receive Stopped Enable When this bit is set with Abnormal Interrupt Summary Enable (BIT 15), Receive Stopped Interrupt is enabled. When this bit is reset, Receive Stopped Interrupt is disabled.
7	RW	0x0	RUE Receive Buffer Unavailable Enable When this bit is set with Abnormal Interrupt Summary Enable (BIT 15), Receive Buffer Unavailable Interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable Interrupt is disabled
6	RW	0x0	RIE Receive Interrupt Enable When this bit is set with Normal Interrupt Summary Enable (BIT 16), Receive Interrupt is enabled. When this bit is reset, Receive Interrupt is disabled.
5	RW	0x0	UNE Underflow Interrupt Enable When this bit is set with Abnormal Interrupt Summary Enable (BIT 15), Transmit Underflow Interrupt is enabled. When this bit is reset, Underflow Interrupt is disabled.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RW	0x0	OVE Overflow Interrupt Enable When this bit is set with Abnormal Interrupt Summary Enable (BIT 15), Receive Overflow Interrupt is enabled. When this bit is reset, Overflow Interrupt is disabled
3	RW	0x0	TJE Transmit Jabber Timeout Enable When this bit is set with Abnormal Interrupt Summary Enable (BIT 15), Transmit Jabber Timeout Interrupt is enabled. When this bit is reset, Transmit Jabber Timeout Interrupt is disabled.
2	RW	0x0	TUE Transmit Buffer Unavailable Enable When this bit is set with Normal Interrupt Summary Enable (BIT 16), Transmit Buffer Unavailable Interrupt is enabled. When this bit is reset, Transmit Buffer Unavailable Interrupt is disabled.
1	RW	0x0	TSE Transmit Stopped Enable When this bit is set with Abnormal Interrupt Summary Enable (BIT 15), Transmission Stopped Interrupt is enabled. When this bit is reset, Transmission Stopped Interrupt is disabled.
0	RW	0x0	TIE Transmit Interrupt Enable When this bit is set with Normal Interrupt Summary Enable (BIT 16), Transmit Interrupt is enabled. When this bit is reset, Transmit Interrupt is disabled.

**GMAC\_OVERFLOW\_CNT**

Address: Operational Base + offset (0x1020)

Missed Frame and Buffer Overflow Counter Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:29	RO	0x0	reserved
28	RC	0x0	FIFO_overflow_bit Overflow bit for FIFO Overflow Counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
27:17	RC	0x000	Frame_miss_number Indicates the number of frames missed by the application This counter is incremented each time the MTL asserts the sideband signal mtl_rxoverflow_o. The counter is cleared when this register is read with mci_be_i[2] at 1'b1.
16	RC	0x0	Miss_frame_overflow_bit Overflow bit for Missed Frame Counter
15:0	RC	0x0000	Frame_miss_number_2 Indicates the number of frames missed by the controller due to the Host Receive Buffer being unavailable. This counter is incremented each time the DMA discards an incoming frame. The counter is cleared when this register is read with mci_be_i[0] at 1'b1.

**GMAC\_REC\_INT\_WDT\_TIMER**

Address: Operational Base + offset (0x1024)

Receive Interrupt Watchdog Timer Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x00	RIWT RI Watchdog Timer count Indicates the number of system clock cycles multiplied by 256 for which the watchdog timer is set. The watchdog timer gets triggered with the programmed value after the RxDMA completes the transfer of a frame for which the RI status bit is not set due to the setting in the corresponding descriptor RDES1[31]. When the watch-dog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when RI bit is set high due to automatic setting of RI as per RDES1[31] of any received frame.

**GMAC\_AXI\_BUS\_MODE**

Address: Operational Base + offset (0x1028)

AXI Bus Mode Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31	RW	0x0	<p>EN_LPI Enable LPI (Low Power Interface) When set to 1, enable the LPI (Low Power Interface) supported by the GMAC and accepts the LPI request from the AXI System Clock controller. When set to 0, disables the Low Power Mode and always denies the LPI request from the AXI System Clock controller.</p>
30	RW	0x0	<p>UNLCK_ON_MGK_RWK Unlock on Magic Packet or Remote Wake Up When set to 1, enables it to request coming out of Low Power mode only when Magic Packet or Remote Wake Up Packet is received. When set to 0, enables it requests to come out of Low Power mode when any frame is received.</p>
29:22	RO	0x0	reserved
21:20	RW	0x1	<p>WR_OSR_LMT AXI Maximum Write Out Standing Request Limit This value limits the maximum outstanding request on the AXI write interface. Maximum outstanding requests = WR_OSR_LMT+1</p>
19:18	RO	0x0	reserved
17:16	RW	0x1	<p>RD_OSR_LMT AXI Maximum Read Out Standing Request Limit This value limits the maximum outstanding request on the AXI read interface. Maximum outstanding requests = RD_OSR_LMT+1</p>
15:13	RO	0x0	reserved
12	RO	0x0	<p>AXI_AAL Address-Aligned Beats This bit is read-only bit and reflects the AAL bit Register0 (register GMAC_BUS_MODE[25]). When this bit set to 1, it performs address-aligned burst transfers on both read and write channels.</p>
11:4	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	BLEN16 AXI Burst Length 16 When this bit is set to 1, or when UNDEF is set to 1, it is allowed to select a burst length of 16.
2	RW	0x0	BLEN8 AXI Burst Length 8 When this bit is set to 1, or when UNDEF is set to 1, it is allowed to select a burst length of 8.
1	RW	0x0	BLEN4 AXI Burst Length 4 When this bit is set to 1, or when UNDEF is set to 1, it is allowed to select a burst length of 4.
0	RO	0x1	UNDEF AXI Undefined Burst Length This bit is read-only bit and indicates the complement (invert) value of FB bit in register GMAC_BUS_MODE[16]. When this bit is set to 1, it is allowed to perform any burst length equal to or below the maximum allowed burst length as programmed in bits[7:1]; When this bit is set to 0, the it is allowed to perform only fixed burst lengths as indicated by BLEN256/128/64/32/16/8/4, or a burst length of 1.

**GMAC\_AXI\_STATUS**

Address: Operational Base + offset (0x102c)

AXI Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RO	0x0	RD_CH_STA When high, it indicates that AXI Master's read channel is active and transferring data.
0	RO	0x0	WR_CH_STA When high, it indicates that AXI Master's write channel is active and transferring data.

**GMAC\_CUR\_HOST\_TX\_DESC**

Address: Operational Base + offset (0x1048)

Current Host Transmit Descriptor Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HTDAP Host Transmit Descriptor Address Pointer Cleared on Reset. Pointer updated by DMA during operation.

**GMAC\_CUR\_HOST\_RX\_DESC**

Address: Operational Base + offset (0x104c)

Current Host Receive Descriptor Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HRDAP Host Receive Descriptor Address Pointer Cleared on Reset. Pointer updated by DMA during operation.

**GMAC\_CUR\_HOST\_TX\_Buf\_ADDR**

Address: Operational Base + offset (0x1050)

Current Host Transmit Buffer Address Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HTBAP Host Transmit Buffer Address Pointer Cleared on Reset. Pointer updated by DMA during operation.

**GMAC\_CUR\_HOST\_RX\_BUF\_ADDR**

Address: Operational Base + offset (0x1054)

Current Host Receive Buffer Adderss Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	HRBAP Host Receive Buffer Address Pointer Cleared on Reset. Pointer updated by DMA during operation.

**GMAC\_HW\_FEA\_REG**

Address: Operational Base + offset (0x1058)

The presence of the optional features/functions of the core Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	Reserved
24	RO	0x0	Alternate (Enhanced Descriptor)
23:20	RO	0x0	Reserved
19	RO	0x1	RxFIFO > 2048 Bytes
18	RO	0x1	IP Checksum Offload (Type 2) in Rx
17	RO	0x0	IP Checksum Offload (Type 1) in Rx
16	RO	0x1	Checksum Offload in Tx
15:14	RO	0x0	Reserved
13	RO	0x0	IEEE 1588-2008 Advanced Time-Stamp

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
12	RO	0x0	IEEE 1588-2002 Time-Stamp
11	RO	0x1	RMON module
10	RO	0x1	PMT Magic Packet
9	RO	0x1	PMT Remote Wakeup
8	RO	0x1	SMA (MDIO) Interface
7	RO	0x0	Reserved
6	RO	0x0	PCS registers (TBI/SGMII/RTBI PHY interface)
5	RO	0x0	Multiple MAC Address Registers
4	RO	0x1	HASH Filter
3	RO	0x0	Reserved
2	RO	0x1	Half-Duplex support
1	RO	0x1	1000 Mbps support
0	RO	0x1	10/100 Mbps support

## 41.5 Interface Description

Table 41-1 RMII Interface Description

<b>Module pin name</b>	<b>Direction</b>	<b>Pad name</b>	<b>IOMUX</b>
RMII interface			
mac_clk	I/O	GPIO4_A[3]	GRF_GPIO4AL_IOMUX[14:12]=3'b011
mac_txen	O	GPIO4_A[4]	GRF_GPIO4AH_IOMUX[2:0]=3'b011
mac_txd1	O	GPIO3_D[5]	GRF_GPIO3DH_IOMUX[6:4]=3'b011
mac_txd0	O	GPIO3_D[4]	GRF_GPIO3DH_IOMUX[2:0]=3'b011
mac_rxdv	I	GPIO4_A[1]	GRF_GPIO4AL_IOMUX[6:4]=3'b011
mac_rxer	I	GPIO4_A[2]	GRF_GPIO4AL_IOMUX[10:8]=3'b011
mac_rxd1	I	GPIO3_D[7]	GRF_GPIO3DH_IOMUX[14:12]=3'b011
mac_rxd0	I	GPIO3_D[6]	GRF_GPIO3DH_IOMUX[10:8]=3'b011
Management interface			
mac_mdio	I/O	GPIO4_A[5]	GRF_GPIO4AH_IOMUX[5:4]=2'b11
mac_mdc	O	GPIO4_A[0]	GRF_GPIO4AL_IOMUX[1:0]=2'b11

Table 41-2 RGMII Interface Description

<b>Module pin name</b>	<b>Direction</b>	<b>Pad name</b>	<b>IOMUX</b>
RGMII/RMII interface			
mac_clk	I/O	GPIO4_A[3]	GRF_GPIO4AL_IOMUX[14:12]=3'b011
mac_txclk	O	GPIO4_B[1]	GRF_GPIO4BL_IOMUX[6:4]=3'b011
mac_txen	O	GPIO4_A[4]	GRF_GPIO4AH_IOMUX[2:0]=3'b011
mac_txd3	O	GPIO3_D[1]	GRF_GPIO3DL_IOMUX[6:4]=3'b011
mac_txd2	O	GPIO3_D[0]	GRF_GPIO3DL_IOMUX[2:0]=3'b011
mac_txd1	O	GPIO3_D[5]	GRF_GPIO3DH_IOMUX[6:4]=3'b011
mac_txd0	O	GPIO3_D[4]	GRF_GPIO3DH_IOMUX[2:0]=3'b011
mac_rxclk	I	GPIO4_A[6]	GRF_GPIO4AH_IOMUX[10:8]=3'b011
mac_rxdv	I	GPIO4_A[1]	GRF_GPIO4AL_IOMUX[6:4]=3'b011
mac_rxd3	I	GPIO3_D[3]	GRF_GPIO3DL_IOMUX[14:12]=3'b011
mac_rxd2	I	GPIO3_D[2]	GRF_GPIO3DL_IOMUX[10:8]=3'b011
mac_rxd1	I	GPIO3_D[7]	GRF_GPIO3DH_IOMUX[14:12]=3'b011
mac_rxd0	I	GPIO3_D[6]	GRF_GPIO3DH_IOMUX[10:8]=3'b011
mac_crs	I	GPIO4_A[7]	GRF_GPIO4AH_IOMUX[14:12]=3'b011

mac_col	I	GPIO4_B[0]	GRF_GPIO4BL_IOMUX[2:0]=3'b011
Management interface			
mac_mdio	I/O	GPIO4_A[5]	GRF_GPIO4AH_IOMUX[5:4]=2'b11
mac_mdc	O	GPIO4_A[0]	GRF_GPIO4AL_IOMUX[1:0]=2'b11

## 41.6 Application Notes

### 41.6.1 Descriptors

The DMA in GMAC can communicate with Host driver through descriptor lists and data buffers. The DMA transfers data frames received by the core to the Receive Buffer in the Host memory, and Transmit data frames from the Transmit Buffer in the Host memory. Descriptors that reside in the Host memory act as pointers to these buffers.

There are two descriptor lists; one for reception, and one for transmission. The base address of each list is written into DMA Registers RX\_DESC\_LIST\_ADDR and TX\_DESC\_LIST\_ADDR, respectively. A descriptor list is forward linked (either implicitly or explicitly). The last descriptor may point back to the first entry to create a ring structure. Explicit chaining of descriptors is accomplished by setting the second address chained in both Receive and Transmit descriptors (RDES1[24] and TDES1[24]). The descriptor lists resides in the Host physical memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the Host physical memory space, and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data, buffer status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA will skip to the next frame buffer when end-of-frame is detected. Data chaining can be enabled or disabled

The descriptor ring and chain structure is shown in following figure.

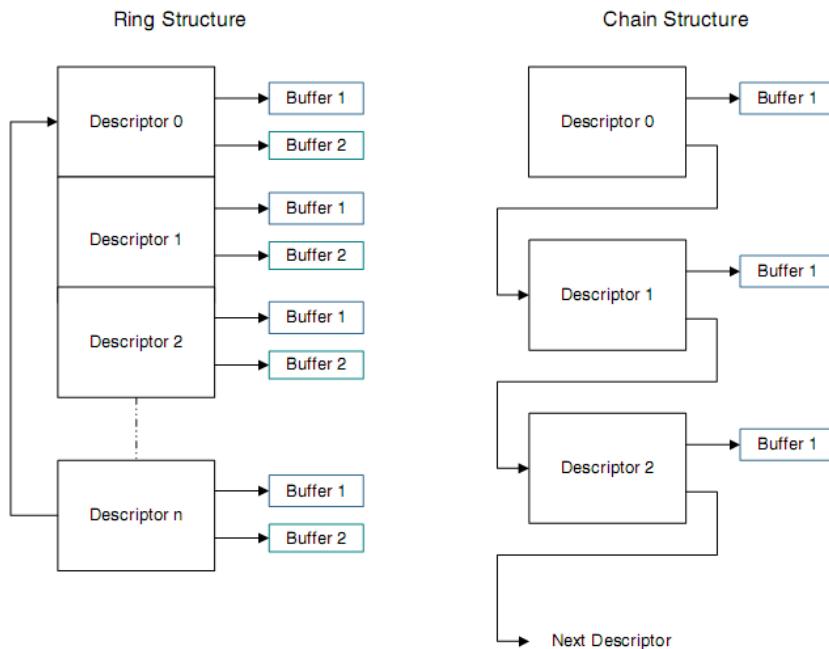


Fig. 41-10 Descriptor Ring and Chain Structure

Each descriptor contains two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory management schemes. The descriptor addresses must be aligned to the bus width used (Word/Dword/Lword for 32/64/128-bit buses).

	63	55	47	39	31	23	15	7	0
DES1-DES0	Control Bits [9:0]	Byte Count Buffer2 [10:0]	Byte Count Buffer1[10:0]	O W N	Status [30:0]				
DES3-DES2		Buffer2 Address [31:0] / Next Descriptor Address [31:0]			Buffer1 Address[31:0]				

Fig. 41-11 Rx/Tx Descriptors definition

#### 41.6.2 Receive Descriptor

The GMAC Subsystem requires at least two descriptors when receiving a frame. The Receive state machine of the DMA always attempts to acquire an extra descriptor in anticipation of an incoming frame. (The size of the incoming frame is unknown). Before the RxDMA closes a descriptor, it will attempt to acquire the next descriptor even if no frames are received.

In a single descriptor (receive) system, the subsystem will generate a descriptor error if the receive buffer is unable to accommodate the incoming frame and the next descriptor is not owned by the DMA. Thus, the Host is forced to increase either its descriptor pool or the buffer size. Otherwise, the subsystem starts dropping all incoming frames.

##### Receive Descriptor 0 (RDES0)

RDES0 contains the received frame status, the frame length, and the descriptor ownership information.

Table 41-3 Receive Descriptor 0

Bit	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA of the GMAC Subsystem. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.
30	AFM: Destination Address Filter Fail When set, this bit indicates a frame that failed in the DA Filter in the GMAC Core.
29:16	FL: Frame Length These bits indicate the byte length of the received frame that was transferred to host memory (including CRC). This field is valid when Last Descriptor (RDES0[8]) is set and either the Descriptor Error (RDES0[14]) or Overflow Error bits are reset. The frame length also includes the two bytes appended to the Ethernet frame when IP checksum calculation (Type 1) is enabled and the received frame is not a MAC control frame. This field is valid when Last Descriptor (RDES0[8]) is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current frame.
15	ES: Error Summary Indicates the logical OR of the following bits: <ul style="list-style-type: none"> <li>• RDES0[0]: Payload Checksum Error</li> <li>• RDES0[1]: CRC Error</li> <li>• RDES0[3]: Receive Error</li> <li>• RDES0[4]: Watchdog Timeout</li> <li>• RDES0[6]: Late Collision</li> <li>• RDES0[7]: IPC Checksum</li> <li>• RDES0[11]: Overflow Error</li> <li>• RDES0[14]: Descriptor Error</li> </ul> This field is valid only when the Last Descriptor (RDES0[8]) is set.
14	DE: Descriptor Error When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next

	Descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set
13	SAF: Source Address Filter Fail When set, this bit indicates that the SA field of frame failed the SA Filter in the GMAC Core.
12	LE: Length Error When set, this bit indicates that the actual length of the frame received and that the Length/ Type field does not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset. Length error status is not valid when CRC error is present.
11	OE: Overflow Error When set, this bit indicates that the received frame was damaged due to buffer overflow.
10	VLAN: VLAN Tag When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the GMACCore.
9	FS: First Descriptor When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.
8	LS: Last Descriptor When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame.
7	IPC Checksum Error/Giant Frame When IP Checksum Engine is enabled, this bit, when set, indicates that the 16-bit IPv4 Header checksum calculated by the core did not match the received checksum bytes. The Error Summary bit[15] is NOT set when this bit is set in this mode.
6	LC: Late Collision When set, this bit indicates that a late collision has occurred while receiving the frame in Half-Duplex mode.
5	FT: Frame Type When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 16'h0600). When this bit is reset, it indicates that the received frame is an IEEE802.3 frame. This bit is not valid for Runt frames less than 14 bytes.
4	RWT: Receive Watchdog Timeout When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.
3	RE: Receive Error When set, this bit indicates that the gmii_rxer_i signal is asserted while gmii_rxrdv_i is asserted during frame reception. This error also includes carrier extension error in GMII and Half-duplex mode. Error can be of less/no extension, or error ( $rxd \neq 0f$ ) during extension.
2	DE: Dribble Bit Error When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in MII Mode.
1	CE: CRC Error When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set.
0	Rx MAC Address/Payload Checksum Error When set, this bit indicates that the Rx MAC Address registers value (1 to 15) matched the frame's DA field. When reset, this bit indicates that the Rx MAC Address Register 0 value matched the DA field. If Full Checksum Offload Engine is enabled, this bit, when set, indicates the TCP, UDP, or ICMP checksum the core calculated does not match the received encapsulated TCP, UDP, or ICMP segment's Checksum field. This bit is also set when the received number

	of payload bytes does not match the value indicated in the Length field of the encapsulated IPv4 or IPv6 datagram in the received Ethernet frame.
--	---

### Receive Descriptor 1 (RDES1)

RDES1 contains the buffer sizes and other bits that control the descriptor chain/ring.

Table 41-4 Receive Descriptor 1

Bit	Description
31	Disable Interrupt on Completion When set, this bit will prevent the setting of the RI (CSR5[6]) bit of the GMAC_STATUS Register for the received frame that ends in the buffer pointed to by this descriptor. This, in turn, will disable the assertion of the interrupt to Host due to RI for that frame.
30:26	Reserved.
25	RER: Receive End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a Descriptor Ring.
24	RCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When RDES1[24] is set, RBS2 (RDES1[21-11]) is a “don’t care” value. RDES1[25] takes precedence over RDES1[24].
23:22	Reserved.
21:11	RBS2: Receive Buffer 2 Size These bits indicate the second data buffer size in bytes. The buffer size must be a multiple of 8 depending upon the bus widths (64), even if the value of RDES3 (buffer2 address pointer) is not aligned to bus width. In the case where the buffer size is not a multiple of 8, the resulting behavior is undefined. This field is not valid if RDES1[24] is set.
10:0	RBS1: Receive Buffer 1 Size Indicates the first data buffer size in bytes. The buffer size must be a multiple of 8 depending upon the bus widths (64), even if the value of RDES2 (buffer1 address pointer) is not aligned. In the case where the buffer size is not a multiple of 8, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (Bit 24).

### Receive Descriptor 2 (RDES2)

RDES2 contains the address pointer to the first data buffer in the descriptor.

Table 41-5 Receive Descriptor 2

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. Note that the DMA performs a write operation with the RDES2[2:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[2:0] (corresponding to bus width of 64) if the address pointer is to a buffer where the middle or last part of the frame is stored.

### Receive Descriptor 3 (RDES3)

RDES3 contains the address pointer either to the second data buffer in the descriptor or to the next descriptor.

Table 41-6 Receive Descriptor 3

Bit	Description
31:0	<p>Buffer 2 Address Pointer (Next Descriptor Address)  These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present.</p> <p>If RDES1[24] is set, the buffer (Next Descriptor) address pointer must be bus width-aligned (RDES3[2:0] = 0, corresponding to a bus width of 64. LSBs are ignored internally.) However, when RDES1[24] is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3[2:0] (corresponding to a bus width of 64) if the address pointer is to a buffer where the middle or last part of the frame is stored.</p>

#### 41.6.3 Transmit Descriptor

The descriptor addresses must be aligned to the bus width used (64). Each descriptor is provided with two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory-management schemes.

##### Transmit Descriptor 0 (TDES0)

TDES0 contains the transmitted frame status and the descriptor ownership information.

Table 41-7 Transmit Descriptor 0

Bit	Description
31	<p>OWN: Own Bit  When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are empty. The ownership bit of the First Descriptor of the frame should be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.</p>
30:17	Reserved.
16	<p>IHE: IP Header Error  When set, this bit indicates that the Checksum Offload engine detected an IP header error and consequently did not modify the transmitted frame for any checksum insertion.</p>
15	<p>ES: Error Summary  Indicates the logical OR of the following bits:</p> <ul style="list-style-type: none"> <li>• TDES0[14]: Jabber Timeout</li> <li>• TDES0[13]: Frame Flush</li> <li>• TDES0[11]: Loss of Carrier</li> <li>• TDES0[10]: No Carrier</li> <li>• TDES0[9]: Late Collision</li> <li>• TDES0[8]: Excessive Collision</li> <li>• TDES0[2]: Excessive Deferral</li> <li>• TDES0[1]: Underflow Error</li> </ul>

14	JT: Jabber Timeout When set, this bit indicates the GMAC transmitter has experienced a jabber time-out.
13	FF: FrameFlushed When set, this bit indicates that the DMA/MTL flushed the frame due to a SW flush command given by the CPU.
12	PCE: Payload Checksum Error This bit, when set, indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either due to insufficient bytes, as indicated by the IP Header's Payload Length field, or the MTL starting to forward the frame to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet frame being transmitted: to avoid deadlock, the MTL starts forwarding the frame when the FIFO is full, even in Store-and-Forward mode.
11	LC: Loss of Carrier When set, this bit indicates that Loss of Carrier occurred during frame transmission. This is valid only for the frames transmitted without collision and when the GMAC operates in Half-Duplex Mode.
10	NC: No Carrier When set, this bit indicates that the carrier sense signal from the PHY was not asserted during transmission.
9	LC: Late Collision When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte times including Preamble in RMII Mode and 512 byte times including Preamble and Carrier Extension in RGMII Mode). Not valid if Underflow Error is set.
8	EC: Excessive Collision When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the DR (Disable Retry) bit in the GMAC Configuration Register is set, this bit is set after the first collision and the transmission of the frame is aborted.
7	VF: VLAN Frame When set, this bit indicates that the transmitted frame was a VLAN-type frame.
6:3	CC: Collision Count This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the Excessive Collisions bit (TDES0[8]) is set.
2	ED: Excessive Deferral When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000-Mbps mode) if the Deferral Check (DC) bit is set high in the GMAC Control Register.
1	UF: Underflow Error When set, this bit indicates that the GMAC aborted the frame because data arrived late from the Host memory. Underflow Error indicates that the DMA encountered an empty Transmit Buffer while transmitting the frame. The transmission process enters the suspended state and sets both Transmit Underflow (Register GMAC_STATUS[5]) and Transmit Interrupt (Register GMAC_STATUS [0]).
0	DB: Deferred Bit When set, this bit indicates that the GMAC defers before transmission because of the presence of carrier. This bit is valid only in Half-Duplex mode.

### Transmit Descriptor 1 (TDES1)

TDES1 contains the buffer sizes and other bits which control the descriptor chain/ring and the

frame being transferred.

Table 41-8 Transmit Descriptor 1

Bit	Description
31	IC: Interrupt on Completion When set, this bit sets Transmit Interrupt (Register 5[0]) after the present frame has been transmitted.
30	LS: Last Segment When set, this bit indicates that the buffer contains the last segment of the frame.
29	FS: First Segment When set, this bit indicates that the buffer contains the first segment of a frame.
28:27	CIC: Checksum Insertion Control These bits control the insertion of checksums in Ethernet frames that encapsulate TCP, UDP, or ICMP over IPv4 or IPv6 as described below. <ul style="list-style-type: none"> <li>• 2'b00: Do nothing. Checksum Engine is bypassed</li> <li>• 2'b01: Insert IPv4 header checksum. Use this value to insert IPv4 header checksum when the frame encapsulates an IPv4 datagram.</li> <li>• 2'b10: Insert TCP/UDP/ICMP checksum. The checksum is calculated over the TCP, UDP, or ICMP segment only and the TCP, UDP, or ICMP pseudo-header checksum is assumed to be present in the corresponding input frame's Checksum field. An IPv4 header checksum is also inserted if the encapsulated datagram conforms to IPv4.</li> <li>• 2'b11: Insert a TCP/UDP/ICMP checksum that is fully calculated in this engine. In other words, the TCP, UDP, or ICMP pseudo-header is included in the checksum calculation, and the input frame's corresponding Checksum field has an all-zero value. An IPv4 Header checksum is also inserted if the encapsulated datagram conforms to IPv4.</li> </ul> The Checksum engine detects whether the TCP, UDP, or ICMP segment is encapsulated in IPv4 or IPv6 and processes its data accordingly.
26	DC: Disable CRC When set, the GMAC does not append the Cyclic Redundancy Check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES1[29]).
25	TER: Transmit End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The returns to the base address of the list, creating a descriptor ring.
24	TCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES1[24] is set, TBS2 (TDES1[21-11]) are "don't care" values. TDES1[25] takes precedence over TDES1[24].
23	DP: Disable Padding When set, the GMAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes and the CRC field is added despite the state of the DC (TDES1[26]) bit. This is valid only when the first segment (TDES1[29]) is set.
22	Reserved.
21:11	TBS2: Transmit Buffer 2 Size These bits indicate the Second Data Buffer in bytes. This field is not valid if TDES1[24] is set.
10:0	TBS1: Transmit Buffer 1 Size These bits indicate the First Data Buffer byte size. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of TCH (Bit 24).

## Transmit Descriptor 2 (TDES2)

TDES2 contains the address pointer to the first buffer of the descriptor.

Table 41-9 Transmit Descriptor 2

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment.

### Transmit Descriptor 3 (TDES3)

TDES3 contains the address pointer either to the second buffer of the descriptor or the next descriptor.

Table 41-10 Transmit Descriptor 3

Bit	Description
31:0	Buffer 2 Address Pointer (Next Descriptor Address) Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES1[24] is set. (LSBs are ignored internally.)

## 41.6.4 Programming Guide

### DMA Initialization – Descriptors

The following operations must be performed to initialize the DMA.

1. Provide a software reset. This will reset all of the GMAC internal registers and logic. (GMAC\_OP\_MODE[0]).
2. Wait for the completion of the reset process (poll GMAC\_OP\_MODE[0], which is only cleared after the reset operation is completed).
3. Program the following fields to initialize the Bus Mode Register by setting values in register GMAC\_BUS\_MODE
  - a. Mixed Burst and AAL
  - b. Fixed burst or undefined burst
  - c. Burst length values and burst mode values.
  - d. Descriptor Length (only valid if Ring Mode is used)
  - e. Tx and Rx DMA Arbitration scheme
4. Program the AXI Interface options in the register GMAC\_BUS\_MODE
  - a. If fixed burst-length is enabled, then select the maximum burst-length possible on the AXI bus (Bits[7:1])
5. A proper descriptor chain for transmit and receive must be created. It should also ensure that the receive descriptors are owned by DMA (bit 31 of descriptor should be set). When OSF mode is used, at least two descriptors are required.
6. Software should create three or more different transmit or receive descriptors in the chain before reusing any of the descriptors.
7. Initialize receive and transmit descriptor list address with the base address of transmit and receive descriptor (register GMAC\_RX\_DESC\_LIST\_ADDR and GMAC\_TX\_DESC\_LIST\_ADDR).
8. Program the following fields to initialize the mode of operation by setting values in register

**GMAC\_OP\_MODE**

- a. Receive and Transmit Store And Forward
  - b. Receive and Transmit Threshold Control (RTC and TTC)
  - c. Hardware Flow Control enable
  - d. Flow Control Activation and De-activation thresholds for MTL Receive and Transmit FIFO (RFA and RFD)
  - e. Error Frame and undersized good frame forwarding enable
  - f. OSF Mode
9. Clear the interrupt requests, by writing to those bits of the status register (interrupt bits only) which are set. For example, by writing 1 into bit 16 - normal interrupt summary will clear this bit (register GMAC\_STATUS).
10. Enable the interrupts by programming the interrupt enable register GMAC\_INT\_ENA.
11. Start the Receive and Transmit DMA by setting SR (bit 1) and ST (bit 13) of the control register GMAC\_OP\_MODE.

**MAC Initialization**

The following MAC Initialization operations can be performed after the DMA initialization sequence. If the MAC Initialization is done before the DMA is set-up, then enable the MAC receiver (last step below) only after the DMA is active. Otherwise, received frames will fill the RxFIFO and overflow. Steps (1) and (2) are to be followed if the TBI/SGMII/RTBI PHY interface is enabled, otherwise follow steps (3) and (4).

1. Program the AN Control register GMAC\_AN\_CTRL to enable Auto-negotiation ANE (bit-12). Setting ELE (bit-14) of this register will enable the PHY to loop back the transmit data.
2. Check the AN Status Register GMAC\_AN\_STATUS for completion of the Auto-negotiation process. ANC (bit-5) should be set, and link status (bit-2), when set, indicates that the link is up.
3. Program the register GMAC\_GMII\_ADDR for controlling the management cycles for external PHY, for example, Physical Layer Address PA (bits 15-11). Also set bit 0 (GMII Busy) for writing into PHY and reading from PHY.
4. Read the 16-bit data of (GMAC\_GMII\_DATA) from the PHY for link up, speed of operation, and mode of operation, by specifying the appropriate address value in register GMAC\_GMII\_ADDR (bits 15-11).
5. Provide the MAC address registers (GMAC\_MAC\_ADDR0\_HI and GMAC\_MAC\_ADDR0\_LO). If more than one MAC address is enabled in your configuration (during configuration in coreConsultant), program them appropriately.
6. If Hash filtering is enabled in your configuration, program the Hash filter register (GMAC\_HASH\_TAB\_HI and GMAC\_HASH\_TAB\_LO).
7. Program the following fields to set the appropriate filters for the incoming frames in register GMAC\_MAC\_FRM\_FILT
  - a. Receive All
  - b. Promiscuous mode
  - c. Hash or Perfect Filter
  - d. Unicast, Multicast, broad cast and control frames filter settings etc.
8. Program the following fields for proper flow control in register GMAC\_FLOW\_CTRL.
  - a. Pause time and other pause frame control bits
  - b. Receive and Transmit Flow control bits
  - c. Flow Control Busy/Backpressure Activate

9. Program the Interrupt Mask register bits, as required, and if applicable, for your configuration.
10. Program the appropriate fields in register GMAC\_MAC\_CONF for example, Inter-frame gap while transmission, jabber disable, etc. Based on the Auto-negotiation you can set the Duplex mode (bit 11), port select (bit 15), etc.
11. Set the bits Transmit enable (TE bit-3) and Receive Enable (RE bit-2) in register GMAC\_MAC\_CONF.

### **Normal Receive and Transmit Operation**

For normal operation, the following steps can be followed.

- For normal transmit and receive interrupts, read the interrupt status. Then poll the descriptors, reading the status of the descriptor owned by the Host (either transmit or receive).
- On completion of the above step, set appropriate values for the descriptors, ensuring that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
- If the descriptors were not owned by the DMA (or no descriptor is available), the DMA will go into SUSPEND state. The transmission or reception can be resumed by freeing the descriptors and issuing a poll demand by writing 0 into the Tx/Rx poll demand register (GMAC\_TX\_POLL\_DEMAND and GMAC\_RX\_POLL\_DEMAND).
- The values of the current host transmitter or receiver descriptor address pointer can be read for the debug process (GMAC\_CUR\_HOST\_TX\_DESC and GMAC\_CUR\_HOST\_RX\_DESC).
- The values of the current host transmit buffer address pointer and receive buffer address pointer can be read for the debug process (GMAC\_CUR\_HOST\_TX\_Buf\_ADDR and GMAC\_CUR\_HOST\_RX\_BUF\_ADDR).

### **Stop and Start Operation**

When the transmission is required to be paused for some time then the following steps can be followed.

1. Disable the Transmit DMA (if applicable), by clearing ST (bit 13) of the control register GMAC\_OP\_MODE.
2. Wait for any previous frame transmissions to complete. This can be checked by reading the appropriate bits of MAC Debug register.
3. Disable the MAC transmitter and MAC receiver by clearing the bits Transmit enable (TE bit-3) and Receive Enable (RE bit-2) in register GMAC\_MAC\_CONF.
4. Disable the Receive DMA (if applicable), after making sure the data in the RX FIFO is transferred to the system memory (by reading the register GMAC\_DEBUG).
5. Make sure both the TX FIFO and RX FIFO are empty.
6. To re-start the operation, start the DMAs first, before enabling the MAC Transmitter and Receiver.

### **41.6.5 Clock Architecture**

In RMII mode, reference clock and TX/RX clock can be from CRU or external OSC as following figure.

The mux select rmii\_speed is GRF\_SOC\_CON1[11].

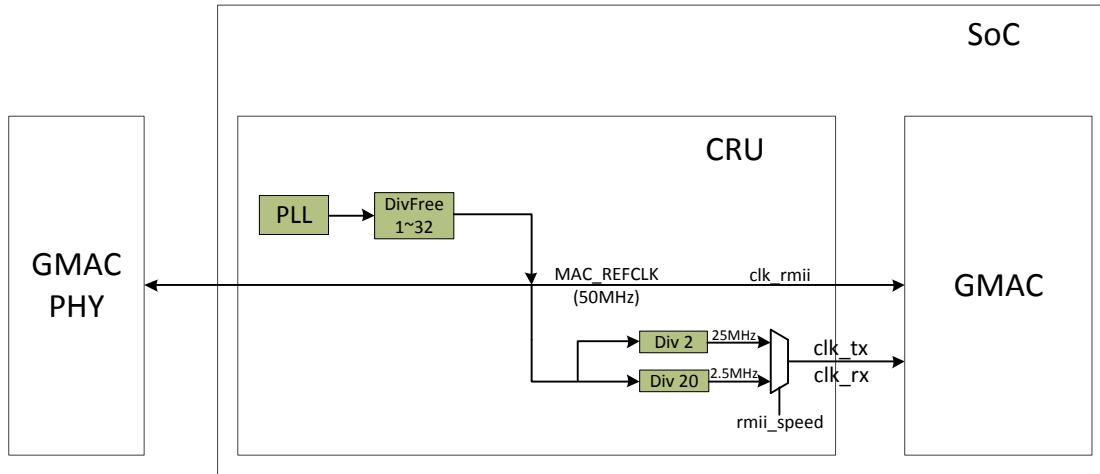


Fig. 41-12 RMII clock architecture when clock source from CRU

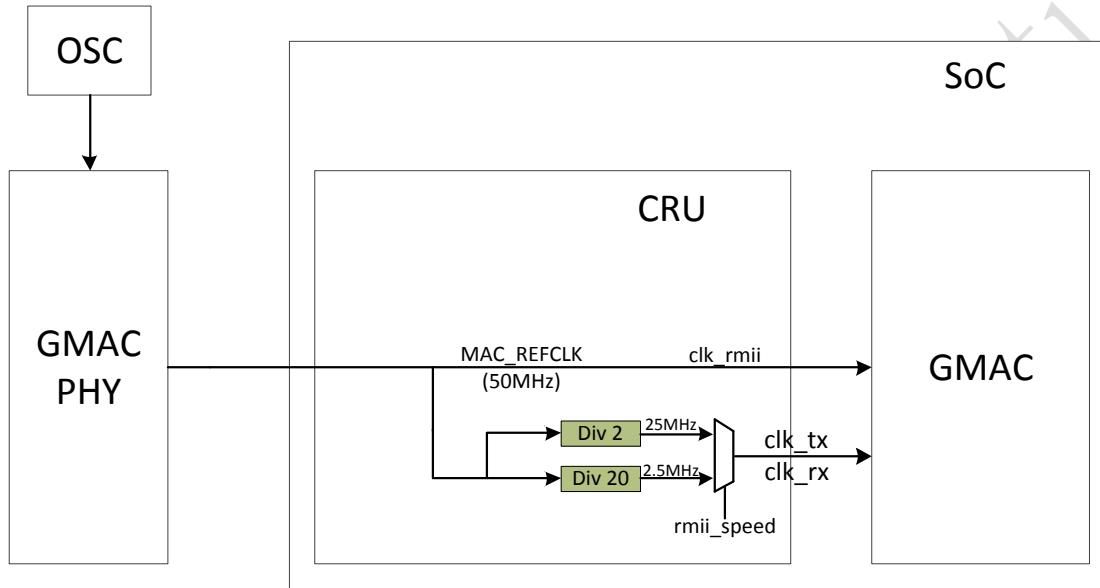


Fig. 41-13 RMII clock architecture when clock source from external OSC

In RGMII mode, clock architecture only supports that TX clock source is from CRU as following figure.

In order to dynamically adjust the timing between TX/RX clock with data, delayline is integrated in TX and RX clock path. Register GRF\_SOC\_CON3[15:14] can enable the delaylines, and GRF\_SOC\_CON3[13:0] is used to determine the delay length. There are 100 delay elements in each delayline, and it totally can adjust about 5.1ns typically.

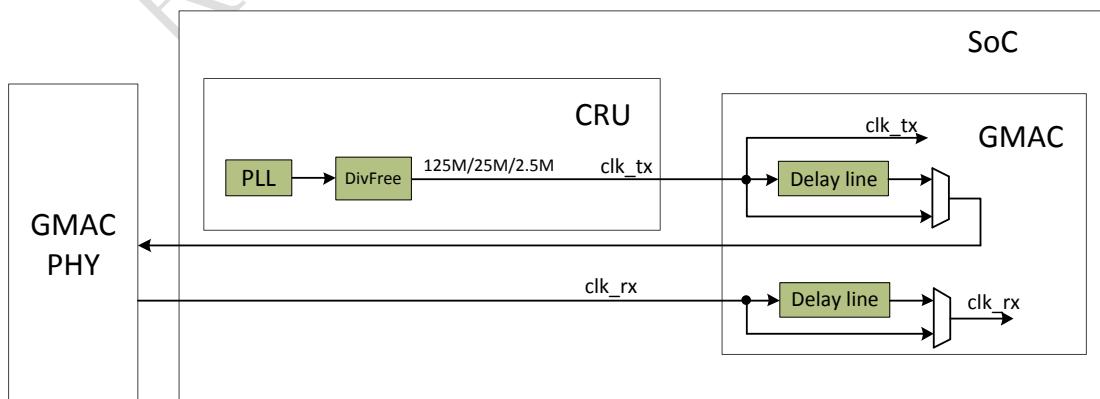


Fig. 41-14 RGMII clock architecture when clock source from CRU

## 41.6.6 Remote Wake-Up Frame Filter Register

The register `wkupfmfilter_reg`, address (028H), loads the Wake-up Frame Filter register. To load values in a Wake-up Frame Filter register, the entire register (`wkupfmfilter_reg`) must be written. The `wkupfmfilter_reg` register is loaded by sequentially loading the eight register values in address (028) for `wkupfmfilter_reg0`, `wkupfmfilter_reg1`, ... `wkupfmfilter_reg7`, respectively. `wkupfmfilter_reg` is read in the same way.

The internal counter to access the appropriate `wkupfmfilter_reg` is incremented when lane3 (or lane 0 in big-endian) is accessed by the CPU. This should be kept in mind if you are accessing these registers in byte or half-word mode.

<code>wkupfmfilter_reg0</code>	Filter 0 Byte Mask											
<code>wkupfmfilter_reg1</code>	Filter 1 Byte Mask											
<code>wkupfmfilter_reg2</code>	Filter 2 Byte Mask											
<code>wkupfmfilter_reg3</code>	Filter 3 Byte Mask											
<code>wkupfmfilter_reg4</code>	RSVD	Filter 3 Command	RSVD	Filter 2 Command	RSVD	Filter 1 Command	RSVD	Filter 0 Command				
<code>wkupfmfilter_reg5</code>	Filter 3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset					
<code>wkupfmfilter_reg6</code>	Filter 1 CRC - 16				Filter 0 CRC - 16							
<code>wkupfmfilter_reg7</code>	Filter 3 CRC - 16				Filter 2 CRC - 16							

Fig. 41-15 Wake-Up Frame Filter Register

### Filter i Byte Mask

This register defines which bytes of the frame are examined by filter i (0, 1, 2, and 3) in order to determine whether or not the frame is a wake-up frame. The MSB (thirty-first bit) must be zero. Bit j [30:0] is the Byte Mask. If bit j (byte number) of the Byte Mask is set, then Filter i Offset + j of the incoming frame is processed by the CRC block; otherwise Filter i Offset + j is ignored.

### Filter i Command

This 4-bit command controls the filter i operation. Bit 3 specifies the address type, defining the pattern's destination address type. When the bit is set, the pattern applies to only multicast frames; when the bit is reset, the pattern applies only to unicast frame. Bit 2 and Bit 1 are reserved. Bit 0 is the enable for filter i; if Bit 0 is not set, filter i is disabled.

### Filter i Offset

This register defines the offset (within the frame) from which the frames are examined by filter i. This 8-bit pattern-offset is the offset for the filter i first byte to examined. The minimum allowed is 12, which refers to the 13th byte of the frame (offset value 0 refers to the first byte of the frame).

### Filter i CRC-16

This register contains the CRC\_16 value calculated from the pattern, as well as the byte mask programmed to the wake-up filter register block.

## 41.6.7 System Consideration During Power-Down

GMAC neither gates nor stops clocks when Power-Down mode is enabled. Power saving by clock gating must be done outside the core by the CRU. The receive data path must be clocked with `clk_rx_i` during Power-Down mode, because it is involved in magic packet/wake-on-LAN frame detection. However, the transmit path and the APB path clocks can be gated off during Power-Down mode.

The pmt interrupt is asserted when a valid wake-up frame is received. This interrupt is generated in the clk\_rx domain.

The recommended power-down and wake-up sequence is as follows.

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. These transmissions can be detected when Transmit Interrupt (TI - Register GMAC\_STATUS[0]) is received.
2. Disable the MAC transmitter and MAC receiver by clearing the appropriate bits in the MAC Configuration register.
3. Wait until the Receive DMA empties all the frames from the Rx FIFO (a software timer may be required).
4. Enable Power-Down mode by appropriately configuring the PMT registers.
5. Enable the MAC Receiver and enter Power-Down mode.
6. Gate the APB and transmit clock inputs to the core (and other relevant clocks in the system) to reduce power and enter Sleep mode.
7. On receiving a valid wake-up frame, the GMAC asserts the pmt interrupt signal and exits Power-Down mode.
8. On receiving the interrupt, the system must enable the APB and transmit clock inputs to the core.
9. Read the register GMAC\_PMT\_CTRL\_STA to clear the interrupt, then enable the other modules in the system and resume normal operation.

#### 41.6.8 GRF Register Summary

GRF Register	Register Description
GRF_SOC_CON1[8:6]	PHY interface select 3'b001: RGMII 3'b100: RMII All others: Reserved
GRF_SOC_CON1[9]	GMAC transmit flow control When set high, instructs the GMAC to transmit PAUSE Control frames in Full-duplex mode. In Half-duplex mode, the GMAC enables the Back-pressure function until this signal is made low again
GRF_SOC_CON1[10]	gmac_speed 1'b1: 100-Mbps 1'b0: 10-Mbps
GRF_SOC_CON1[11]	RMII clock selection 1'b1: 25MHz 1'b0: 2.5MHz
GRF_SOC_CON1[13:12]	RGMII clock selection 2'b00: 125MHz 2'b11: 25MHz 2'b10: 2.5MHz
GRF_SOC_CON1[14]	RMII mode selection 1'b1: RMII mode 1'b0: Reserved
GRF_SOC_CON3[6:0]	RGMII TX clock delayline value
GRF_SOC_CON3[13:7]	RGMII RX clock delayline value
GRF_SOC_CON3[14]	RGMII TX clock delayline enable

	1'b1: enable 1'b0: disable
GRF_SOC_CON3[15]	RGMII RX clock delayline enable 1'b1: enable 1'b0: disable

## Chapter 42 Serial Peripheral Interface (SPI)

### 42.1 Overview

The serial peripheral interface is an APB slave device. A four wire full duplex serial protocol from Motorola. There are four possible combinations for the serial clock phase and polarity. The clock phase (SCPH) determines whether the serial transfer begins with the falling edge of slave select signals or the first edge of the serial clock. The slave select line is held high when the SPI is idle or disabled. This SPI controller can work as either master or slave mode.

SPI Controller supports the following features:

- Support Motorola SPI, TI Synchronous Serial Protocol and National Semiconductor Microwire interface
- Support 32-bit APB bus
- Support two internal 16-bit wide and 32-location deep FIFOs, one for transmitting and the other for receiving serial data
- Support two chip select signals in master mode
- Support 4, 8, 16 bit serial data transfer
- Support configurable interrupt polarity
- Support asynchronous APB bus and SPI clock
- Support master and slave mode
- Support DMA handshake interface and configurable DMA water level
- Support transmit FIFO empty, underflow, receive FIFO full, overflow, interrupt and all interrupts can be masked
- Support configurable water level of transmit FIFO empty and receive FIFO full interrupt
- Support combine interrupt output
- Support up to half of SPI clock frequency transfer in master mode and one sixth of SPI clock frequency transfer in slave mode
- Support full and half duplex mode transfer
- Stop transmitting SCLK if transmit FIFO is empty or receive FIFO is full in master mode
- Support configurable delay from chip select active to SCLK active in master mode
- Support configurable period of chip select inactive between two parallel data in master mode
- Support big and little endian, MSB and LSB first transfer
- Support two 8-bit audio data store together in one 16-bit wide location
- Support sample RXD 0~3 SPI clock cycles later
- Support configurable SCLK polarity and phase
- Support fix and incremental address access to transmit and receive FIFO

### 42.2 Block Diagram

The SPI Controller comprises with:

- AMBA APB interface and DMA Controller Interface
- Transmit and receive FIFO controllers and an FSM controller
- Register block
- Shift control and interrupt

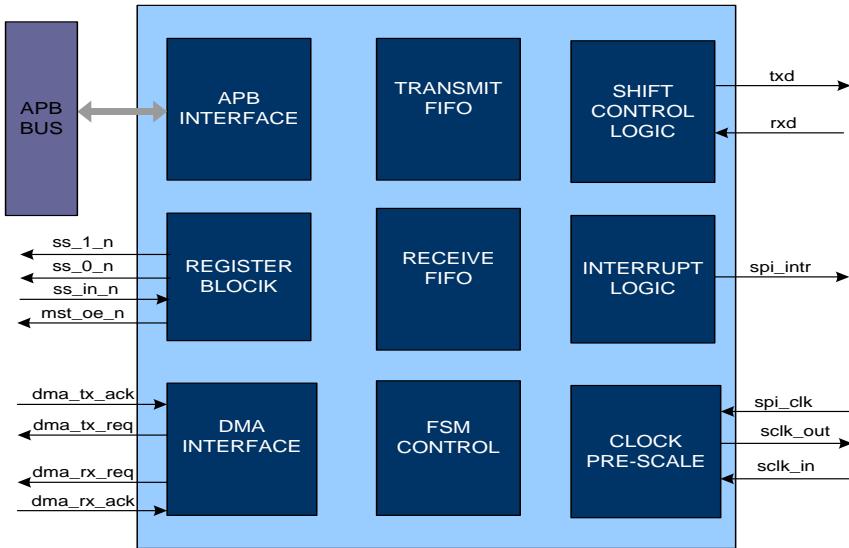


Fig. 42-1 SPI Controller Block diagram

## APB INTERFACE

The host processor accesses data, control, and status information on the SPI through the APB interface. The SPI supports APB data bus widths of 8, 16, and 32 bits.

## DMA INTERFACE

This block has a handshaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA Controller.

## FIFO LOGIC

For transmit and receive transfers, data transmitted from the SPI to the external serial device is written into the transmit FIFO. Data received from the external serial device into the SPI is pushed into the receive FIFO. Both fifos are 32x16bits.

## FSM CONTROL

Control the state's transformation of the design.

## REGISTER BLOCK

All registers in the SPI are addressed at 32-bit boundaries to remain consistent with the AHB bus. Where the physical size of any register is less than 32-bits wide, the upper unused bits of the 32-bit boundary are reserved. Writing to these bits has no effect; reading from these bits returns 0.

## SHIFT CONTROL

Shift control logic shift the data from the transmit fifo or to the receive fifo. This logic automatically right-justifies receive data in the receive FIFO buffer.

## INTERRUPT CONTROL

The SPI supports combined and individual interrupt requests, each of which can be masked. The combined interrupt request is the ORed result of all other SPI interrupts after masking.

## 42.3 Function description

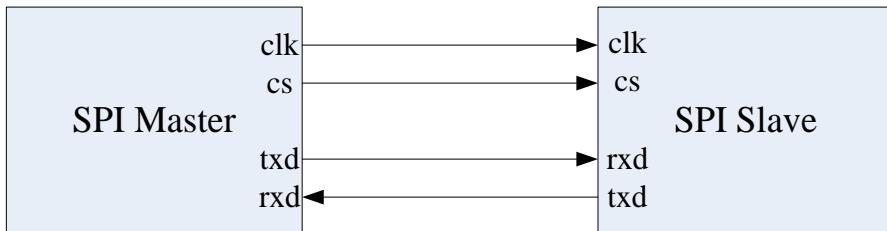


Fig. 42-2 SPI Master and Slave Interconnection

The SPI controller support dynamic switching between master and slave in a system. The diagram show how the SPI controller connects with other SPI devices.

### Operation Modes

The SPI can be configured in the following two fundamental modes of operation: Master Mode when SPI\_CTRLR0 [20] is 1'b0, Slave Mode when SPI\_CTRLR0 [20] is 1'b1.

### Transfer Modes

The SPI operates in the following three modes when transferring data on the serial bus.

#### 1. Transmit and Receive

When SPI\_CTRLR0 [19:18] == 2'b00, both transmit and receive logic are valid.

#### 2. Transmit Only

When SPI\_CTRLR0 [19:18] == 2'b01, the receive data are invalid and should not be stored in the receive FIFO.

#### 3. Receive Only

When SPI\_CTRLR0 [19:18] == 2'b10, the transmit data are invalid.

### Clock Ratios

A summary of the frequency ratio restrictions between the bit-rate clock (sclk\_out / sclk\_in) and the SPI peripheral clock (spi\_clk) are described as,

When SPI Controller works as master, the  $F_{spi\_clk} \geq 2 \times (\text{maximum } F_{sclk\_out})$

When SPI Controller works as slave, the  $F_{spi\_clk} \geq 6 \times (\text{maximum } F_{sclk\_in})$

With the SPI, the clock polarity (SCPOL) configuration parameter determines whether the inactive state of the serial clock is high or low. To transmit data, both SPI peripherals must have identical serial clock phase (SCPH) and clock polarity (SCPOL) values. The data frame can be 4/8/16 bits in length.

When the configuration parameter SCPH = 0, data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock; therefore, valid data must be present on the txd and rxd lines prior to the first serial clock edge. The following two figures show a timing diagram for a single SPI data transfer with SCPH = 0. The serial clock is shown for configuration parameters SCPOL = 0 and SCPO = 1.

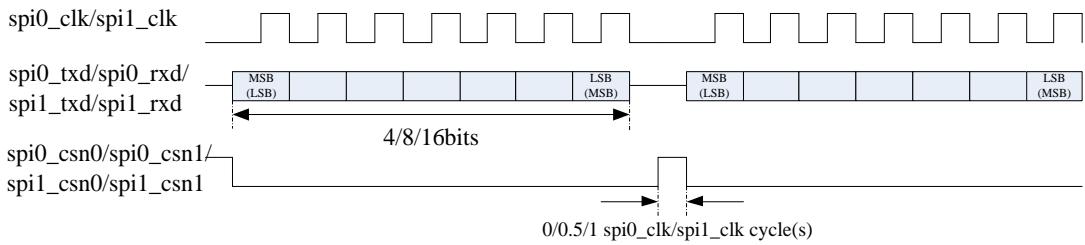


Fig. 42-3 SPI Format (SCPH=0 SCPOL=0)

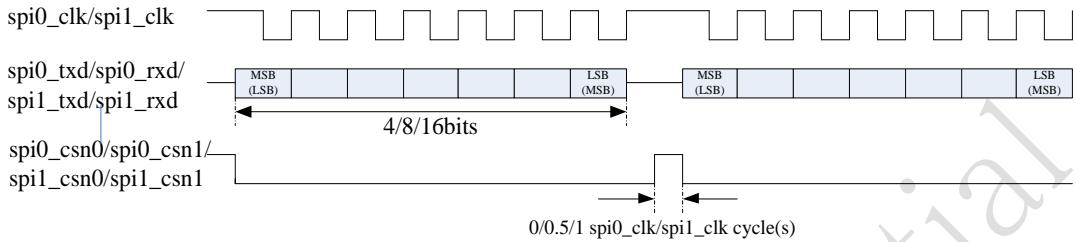


Fig. 42-4 SPI Format (SCPH=0 SCPOL=1)

When the configuration parameter  $\text{SCPH} = 1$ , both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data are propagated by the master and slave peripherals on the leading edge of the serial clock. During continuous data frame transfers, the slave select line may be held active-low until the last bit of the last frame has been captured. The following two figures show the timing diagram for the SPI format when the configuration parameter  $\text{SCPH} = 1$ .

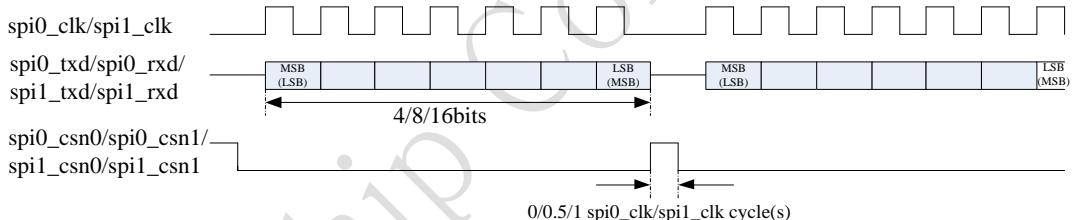


Fig. 42-5 SPI Format (SCPH=1 SCPOL=0)

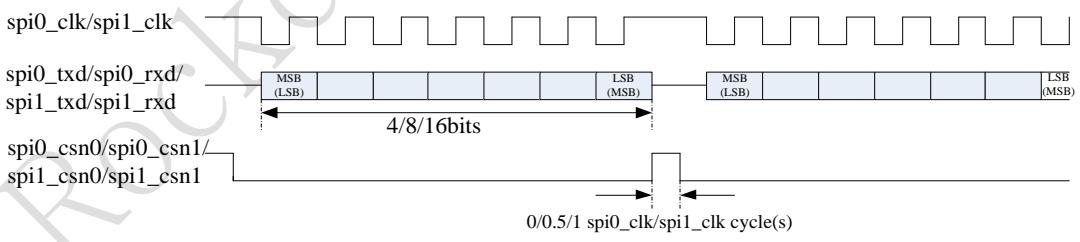


Fig. 42-6 SPI Format (SCPH=1 SCPOL=1)

## 42.4 Register Description

This section describes the control/status registers of the design. Pay attention that there are two SPI controllers in the chip: spi0 & spi1, so the base address in the following register

descriptions can be either spi0 or spi1 base address.

#### 42.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SPI_CTRLR0	0x0000	W	0x00000002	Control Register 0
SPI_CTRLR1	0x0004	W	0x00000000	Control Register 1
SPI_ENR	0x0008	W	0x00000000	SPI Enable
SPI_SER	0x000c	W	0x00000000	Slave Enable Register
SPI_BAUDR	0x0010	W	0x00000000	Baud Rate Select
SPI_TXFTLR	0x0014	W	0x00000000	Transmit FIFO Threshold Level
SPI_RXFTLR	0x0018	W	0x00000000	Receive FIFO Threshold Level
SPI_TXFLR	0x001c	W	0x00000000	Transmit FIFO Level
SPI_RXFLR	0x0020	W	0x00000000	Receive FIFO Level
SPI_SR	0x0024	W	0x0000000c	SPI Status
SPI_IPR	0x0028	W	0x00000000	Interrupt Polarity
SPI_IMR	0x002c	W	0x00000000	Interrupt Mask
SPI_ISR	0x0030	W	0x00000000	Interrupt Status
SPI_RISR	0x0034	W	0x00000001	Raw Interrupt Status
SPI_ICR	0x0038	W	0x00000000	Interrupt Clear
SPI_DMACR	0x003c	W	0x00000000	DMA Control
SPI_DMATDLR	0x0040	W	0x00000000	DMA Transmit Data Level
SPI_DMARDLR	0x0044	W	0x00000000	DMA Receive Data Level
SPI_TXDR	0x0400~0x07fc	W	0x00000000	Transmit FIFO Data
SPI_RXDR	0x0800~0x0bfc	W	0x00000000	Receive FIFO Data

Notes: **Size** : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

#### 42.4.2 Detail Register Description

##### SPI\_CTRLR0

Address: Operational Base + offset (0x0000)

Control Register 0

Bit	Attr	Reset Value	Description
31:22	RO	0x0	reserved
21	RW	0x0	MTM Microwire Transfer Mode Valid when frame format is set to National Semiconductors Microwire. 1'b0: non-sequential transfer 1'b1: sequential transfer

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
20	RW	0x0	OPM Operation Mode 1'b0: Master Mode 1'b1: Slave Mode
19:18	RW	0x0	XFM Transfer Mode 2'b00 :Transmit & Receive 2'b01 : Transmit Only 2'b10 : Receive Only 2'b11 :reserved
17:16	RW	0x0	FRF Frame Format 2'b00: Motorola SPI 2'b01: Texas Instruments SSP 2'b10: National Semiconductors Microwire 2'b11 : Reserved
15:14	RW	0x0	RSD Rxd Sample Delay When SPI is configured as a master, if the rxd data cannot be sampled by the sclk_out edge at the right time, this register should be configured to define the number of the spi_clk cycles after the active sclk_out edge to sample rxd data later when SPI works at high frequency. 2'b00:do not delay 2'b01:1 cycle delay 2'b10:2 cycles delay 2'b11:3 cycles delay
13	RW	0x0	BHT Byte and Halfword Transform Valid when data frame size is 8bit. 1'b0: apb 16bit write/read, spi 8bit write/read 1'b1: apb 8bit write/read, spi 8bit write/read
12	RW	0x0	FBM First Bit Mode 1'b0: first bit is MSB 1'b1: first bit is LSB
11	RW	0x0	EM Endian Mode Serial endian mode can be configured by this bit. Apb endian mode is always little endian. 1'b0: little endian 1'b1: big endian

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
10	RW	0x0	SSD ss_n to sclk_out delay Valid when the frame format is set to Motorola SPI and SPI used as a master. 1'b0: the period between ss_n active and sclk_out active is half sclk_out cycles. 1'b1: the period between ss_n active and sclk_out active is one sclk_out cycle.
9:8	RW	0x0	CSM Chip Select Mode Valid when the frame format is set to Motorola SPI and SPI used as a master. 2'b00: ss_n keep low after every frame data is transferred. 2'b01: ss_n be high for half sclk_out cycles after every frame data is transferred. 2'b10: ss_n be high for one sclk_out cycle after every frame data is transferred. 2'b11: reserved
7	RW	0x0	SCPOL Serial Clock Polarity Valid when the frame format is set to Motorola SPI. 1'b0: Inactive state of serial clock is low 1'b1: Inactive state of serial clock is high
6	RW	0x0	SCPH Serial Clock Phase Valid when the frame format is set to Motorola SPI. 1'b0: Serial clock toggles in middle of first data bit 1'b1: Serial clock toggles at start of first data bit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5:2	RW	0x0	CFS Control Frame Size Selects the length of the control word for the Microwire frame format. 4'b0000~4'b0010: reserved 4'b0011: 4-bit serial data transfer 4'b0100: 5-bit serial data transfer 4'b0101: 6-bit serial data transfer 4'b0110: 7-bit serial data transfer 4'b0111: 8-bit serial data transfer 4'b1000: 9-bit serial data transfer 4'b1001: 10-bit serial data transfer 4'b1010: 11-bit serial data transfer 4'b1011: 12-bit serial data transfer 4'b1100: 13-bit serial data transfer 4'b1101: 14-bit serial data transfer 4'b1110: 15-bit serial data transfer 4'b1111: 16-bit serial data transfer
1:0	RW	0x2	DFS Data Frame Size Selects the data frame length. 2'b00: 4bit data 2'b01: 8bit data 2'b10: 16bit data 2'b11: reserved

**SPI\_CTRLR1**

Address: Operational Base + offset (0x0004)

Control Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	NDM Number of Data Frames When Transfer Mode is receive only, this register field sets the number of data frames to be continuously received by the SPI. The SPI continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables you to receive up to 64 KB of data in a continuous transfer.

**SPI\_ENR**

Address: Operational Base + offset (0x0008)

SPI Enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	ENR SPI Enable Enables and disables all SPI operations. Transmit and receive FIFO buffers are cleared when the device is disabled.

**SPI\_SER**

Address: Operational Base + offset (0x000c)

Slave Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1:0	RW	0x0	SER Slave Select Enable This register is valid only when SPI is configured as a master device.

**SPI\_BAUDR**

Address: Operational Base + offset (0x0010)

Baud Rate Select

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	BAUDR Baud Rate Select SPI Clock Divider. This register is valid only when the SPI is configured as a master device. The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. If the value is 0, the serial output clock (sclk_out) is disabled. The frequency of the sclk_out is derived from the following equation: $F_{sclk\_out} = F_{spi\_clk} / SCKDV$ Where SCKDV is any even value between 2 and 65534. For example: for $F_{spi\_clk} = 3.6864\text{MHz}$ and $SCKDV = 2$ $F_{sclk\_out} = 3.6864/2 = 1.8432\text{MHz}$

**SPI\_TXFTLR**

Address: Operational Base + offset (0x0014)

Transmit FIFO Threshold Level

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RW	0x00	TXFTLR Transmit FIFO Threshold Level When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered.

**SPI\_RXFTLR**

Address: Operational Base + offset (0x0018)

Receive FIFO Threshold Level

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RW	0x00	RXFTLR Receive FIFO Threshold Level When the number of receive FIFO entries is greater than or equal to this value + 1, the receive FIFO full interrupt is triggered.

**SPI\_TXFLR**

Address: Operational Base + offset (0x001c)

Transmit FIFO Level

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RO	0x00	TXFLR Transmit FIFO Level Contains the number of valid data entries in the transmit FIFO.

**SPI\_RXFLR**

Address: Operational Base + offset (0x0020)

Receive FIFO Level

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RO	0x00	RXFLR Receive FIFO Level Contains the number of valid data entries in the receive FIFO.

**SPI\_SR**

Address: Operational Base + offset (0x0024)

SPI Status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RO	0x0	RFF Receive FIFO Full 1'b0: Receive FIFO is not full 1'b1: Receive FIFO is full
3	RO	0x1	RFE Receive FIFO Empty 1'b0: Receive FIFO is not empty 1'b1: Receive FIFO is empty
2	RO	0x1	TFE Transmit FIFO Empty 1'b0: Transmit FIFO is not empty 1'b1: Transmit FIFO is empty
1	RO	0x0	TFF Transmit FIFO Full 1'b0: Transmit FIFO is not full 1'b1: Transmit FIFO is full
0	RO	0x0	BSF SPI Busy Flag When set, indicates that a serial transfer is in progress; when cleared indicates that the SPI is idle or disabled. 1'b0: SPI is idle or disabled 1'b1: SPI is actively transferring data

**SPI\_IPR**

Address: Operational Base + offset (0x0028)

Interrupt Polarity

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	IPR Interrupt Polarity Interrupt Polarity Register 1'b0: Active Interrupt Polarity Level is HIGH 1'b1: Active Interrupt Polarity Level is LOW

**SPI\_IMR**

Address: Operational Base + offset (0x002c)

Interrupt Mask

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4	RW	0x0	RFFIM Receive FIFO Full Interrupt Mask 1'b0: spi_rxf_intr interrupt is masked 1'b1: spi_rxf_intr interrupt is not masked

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	RFOIM Receive FIFO Overflow Interrupt Mask 1'b0: spi_rxo_intr interrupt is masked 1'b1: spi_rxo_intr interrupt is not masked
2	RW	0x0	RFUIM Receive FIFO Underflow Interrupt Mask 1'b0: spi_rxu_intr interrupt is masked 1'b1: spi_rxu_intr interrupt is not masked
1	RW	0x0	TFOIM Transmit FIFO Overflow Interrupt Mask 1'b0: spi_txo_intr interrupt is masked 1'b1: spi_txo_intr interrupt is not masked
0	RW	0x0	TFEIM Transmit FIFO Empty Interrupt Mask 1'b0: spi_txe_intr interrupt is masked 1'b1: spi_txe_intr interrupt is not masked

**SPI\_ISR**

Address: Operational Base + offset (0x0030)

Interrupt Status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4	RO	0x0	RFFIS Receive FIFO Full Interrupt Status 1'b0: spi_rxf_intr interrupt is not active after masking 1'b1: spi_rxf_intr interrupt is full after masking
3	RO	0x0	RFOIS Receive FIFO Overflow Interrupt Status 1'b0: spi_rxo_intr interrupt is not active after masking 1'b1: spi_rxo_intr interrupt is active after masking
2	RO	0x0	RFUIS Receive FIFO Underflow Interrupt Status 1'b0: spi_rxu_intr interrupt is not active after masking 1'b1: spi_rxu_intr interrupt is active after masking

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RO	0x0	TFOIS Transmit FIFO Overflow Interrupt Status 1'b0: spi_txo_intr interrupt is not active after masking 1'b1: spi_txo_intr interrupt is active after masking
0	RO	0x0	TFEIS Transmit FIFO Empty Interrupt Status 1'b0: spi_txe_intr interrupt is not active after masking 1'b1: spi_txe_intr interrupt is active after masking

**SPI\_RISR**

Address: Operational Base + offset (0x0034)

Raw Interrupt Status

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4	RO	0x0	RFFRIS Receive FIFO Full Raw Interrupt Status 1'b0: spi_rxf_intr interrupt is not active prior to masking 1'b1: spi_rxf_intr interrupt is full prior to masking
3	RO	0x0	RFORIS Receive FIFO Overflow Raw Interrupt Status 1'b0: spi_rxo_intr interrupt is not active prior to masking 1'b1: spi_rxo_intr interrupt is active prior to masking
2	RO	0x0	RFURIS Receive FIFO Underflow Raw Interrupt Status 1'b0: spi_rxu_intr interrupt is not active prior to masking 1'b1: spi_rxu_intr interrupt is active prior to masking
1	RO	0x0	TFORIS Transmit FIFO Overflow Raw Interrupt Status 1'b0: spi_txo_intr interrupt is not active prior to masking 1'b1: spi_txo_intr interrupt is active prior to masking

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RO	0x1	TFERIS Transmit FIFO Empty Raw Interrupt Status 1'b0: spi_txe_intr interrupt is not active prior to masking 1'b1: spi_txe_intr interrupt is active prior to masking

**SPI\_ICR**

Address: Operational Base + offset (0x0038)

Interrupt Clear

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3	WO	0x0	CTFOI Clear Transmit FIFO Overflow Interrupt Write 1 to Clear Transmit FIFO Overflow Interrupt
2	WO	0x0	CRFOI Clear Receive FIFO Overflow Interrupt Write 1 to Clear Receive FIFO Overflow Interrupt
1	WO	0x0	CRFUI Clear Receive FIFO Underflow Interrupt Write 1 to Clear Receive FIFO Underflow Interrupt
0	WO	0x0	CCI Clear Combined Interrupt Write 1 to Clear Combined Interrupt

**SPI\_DMACR**

Address: Operational Base + offset (0x003c)

DMA Control

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	TDE Transmit DMA Enable 1'b0: Transmit DMA disabled 1'b1: Transmit DMA enabled
0	RW	0x0	RDE Receive DMA Enable 1'b0: Receive DMA disabled 1'b1: Receive DMA enabled

**SPI\_DMATDLR**

Address: Operational Base + offset (0x0040)

DMA Transmit Data Level

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RW	0x00	TDL Transmit Data Level This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and Transmit DMA Enable (DMACR[1]) = 1.

**SPI\_DMARDLR**

Address: Operational Base + offset (0x0044)

DMA Receive Data Level

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RW	0x00	RDL Receive Data Level This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and Receive DMA Enable(DMACR[0])=1.

**SPI\_TXDR**

Address: Operational Base + offset (0x0400~0x07fc)

Transmit FIFO Data

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	WO	0x0000	TXDR Transmit FIFO Data Register. When it is written to, data are moved into the transmit FIFO.

**SPI\_RXDR**

Address: Operational Base + offset (0x0800~0x0bf0)

Receive FIFO Data

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RO	0x0	reserved
15:0	RW	0x0000	RXDR Receive FIFO Data Register. When the register is read, data in the receive FIFO is accessed.

## 42.5 Interface description

Table 42-1 SPI interface description

Module Pin	Direction	Pad Name	IOMUX Setting
spi0_clk	I/O	IO_SPI0clk_TS0data4_UART4EXPctsn_BBgpio5b4	GRF_GPIO5B_IOM UX[9:8]=01
spi0_csn0	I/O	IO_SPI0csn0_TS0data5_UART4EXPrtsn_BBgpio5b5	GRF_GPIO5B_IOM UX[11:10]=01
spi0_txd	O	IO_SPI0txd_TS0data6_UART4EXPsout_BBgpio5b6	GRF_GPIO5B_IOM UX[13:12]=01
spi0_rxd	I	IO_SPI0rxd_TS0data7_UART4EXPsin_BBgpio5b7	GRF_GPIO5B_IOM UX[15:14]=01
spi0_csn1	O	IO_SPI0csn1_TS0sync_BB gpio5c0	GRF_GPIO5C_IOM UX[1:0]=01
spi1_clk	I/O	IO_ISPshutteren_SPI1clk_GPIO30gpio7b4	GRF_GPIO7B_IOM UX[9:8]=10
spi1_csn0	I/O	IO_ISPflashtrigout_SPI1cs_n0_GPIO30gpio7b5	GRF_GPIO7B_IOM UX[11:10]=10
spi1_rxd	O	IO_ISPprelighttrig_SPI1rxd _GPIO30gpio7b6	GRF_GPIO7B_IOM UX[13:12]=10
spi1_txd	I	IO_ISPshuttertrig_SPI1txd _GPIO30gpio7b7	GRF_GPIO7B_IOM UX[15:14]=10
spi2_clk	I/O	IO_SPI2clk_SCio_GPIO183_0gpio8a6	GRF_GPIO8A_IOM UX[13:12]=01
spi2_csn0	I/O	IO_SPI2csn0_SCDetect_GP IO1830gpio8a7	GRF_GPIO8A_IOM UX[15:14]=01
spi2_rxd	I	IO_SPI2rxd_SCrst_GPIO183_0gpio8b0	GRF_GPIO8B_IOM UX[1:0]=01
spi2_txd	O	IO_SPI2txd_SCclk_GPIO183_0gpio8b1	GRF_GPIO8B_IOM UX[3:2]=01
spi2_csn1	O	IO_SPI2csn1_SCIot1_GPIO183_0gpio8a3	GRF_GPIO8A_IOM UX[7:6]=01

Note: *spi0\_csn1*, *spi1\_csn1*, *spi2\_csn1* can only be used in master mode

## 42.6 Application Notes

### Clock Ratios

A summary of the frequency ratio restrictions between the bit-rate clock (*sclk\_out*/*sclk\_in*) and the SPI peripheral clock (*spi\_clk*) are described as,

When SPI Controller works as master, the  $F_{\text{spi\_clk}} \geq 2 \times (\text{maximum } F_{\text{sclk\_out}})$

When SPI Controller works as slave, the  $F_{\text{spi\_clk}} \geq 6 \times (\text{maximum } F_{\text{sclk\_in}})$

### Master Transfer Flow

When configured as a serial-master device, the SPI initiates and controls all serial transfers. The serial bit-rate clock, generated and controlled by the SPI, is driven out on the *sclk\_out* line. When the SPI is disabled (*SPI\_ENR* = 0), no serial transfers can occur and *sclk\_out* is held in "inactive" state, as defined by the serial protocol under which it operates.

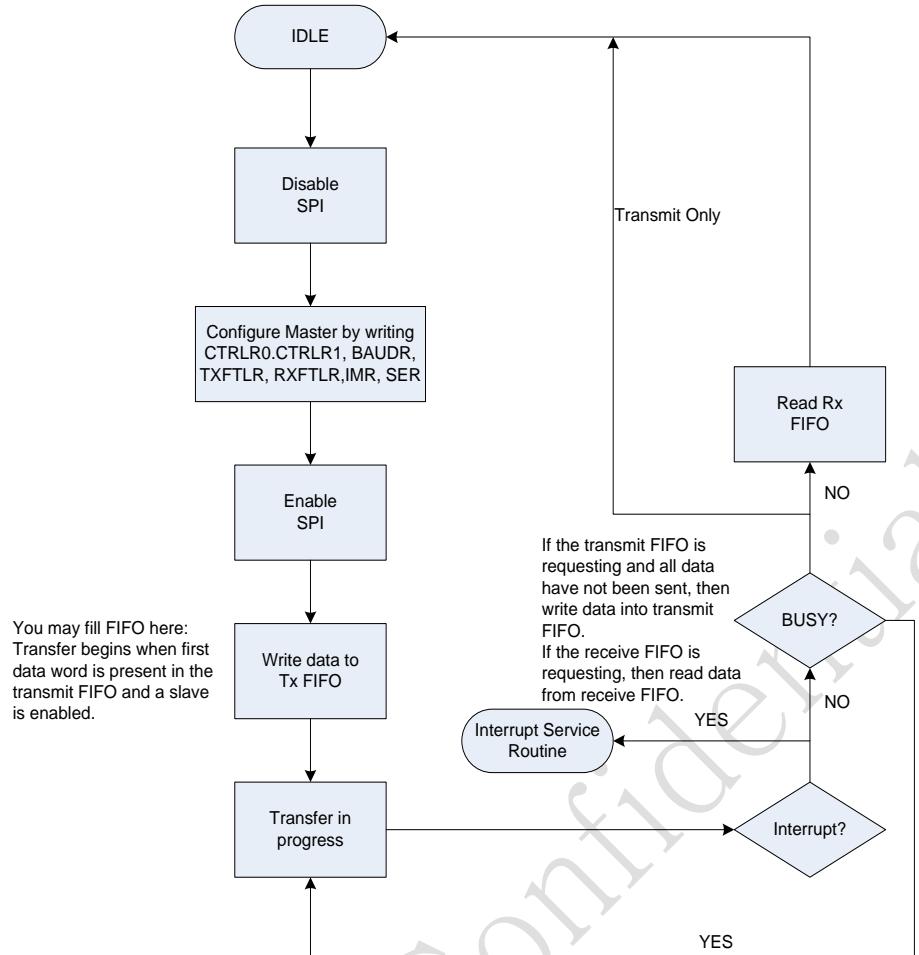


Fig. 42-7 SPI Master transfer flow diagram

### Slave Transfer Flow

When the SPI is configured as a slave device, all serial transfers are initiated and controlled by the serial bus master.

When the SPI serial slave is selected during configuration, it enables its txd data onto the serial bus. All data transfers to and from the serial slave are regulated on the serial clock line (sclk\_in), driven from the serial-master device. Data are propagated from the serial slave on one edge of the serial clock line and sampled on the opposite edge.

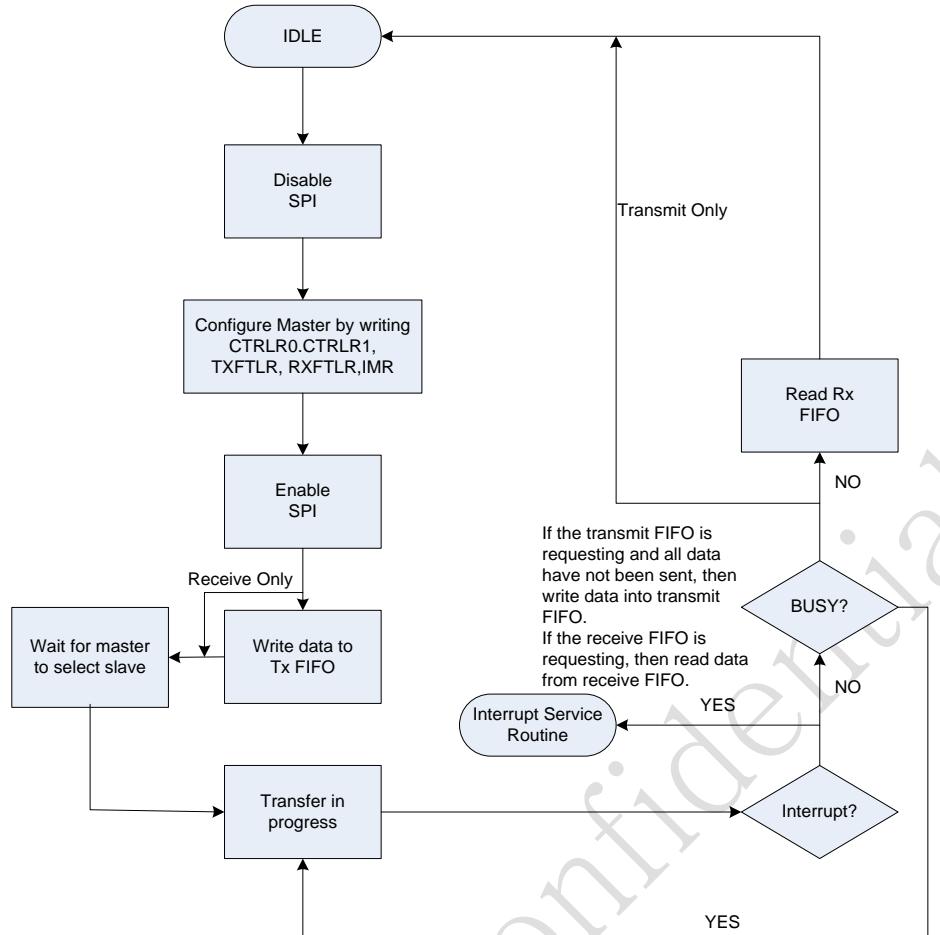


Fig. 42-8 SPI Slave transfer flow diagram

## Chapter 43 PS/2 Controller

### 43.1 Overview

The PS/2 is a bi-directional serial bus protocol that provides an efficient and simple method of information exchange between host and devices. This PS/2 controller supports master mode acting as a bridge between AMBA APB protocol and generic PS/2 bus system.

PS/2 controller supports the following features:

- Support 32bits AMBA2.0 APB bus interface protocol
- Support PS/2 data communication protocol
- Support PS/2 master mode
- Software programmable timing requirement to support max PS/2 clock frequency to 33KHZ
- Support status to be queried for data communication error
- Support interrupt mode for data communication finish
- Support timeout mechanism for data communication
- Support interrupt mode for data communication timeout

### 43.2 Block Diagram

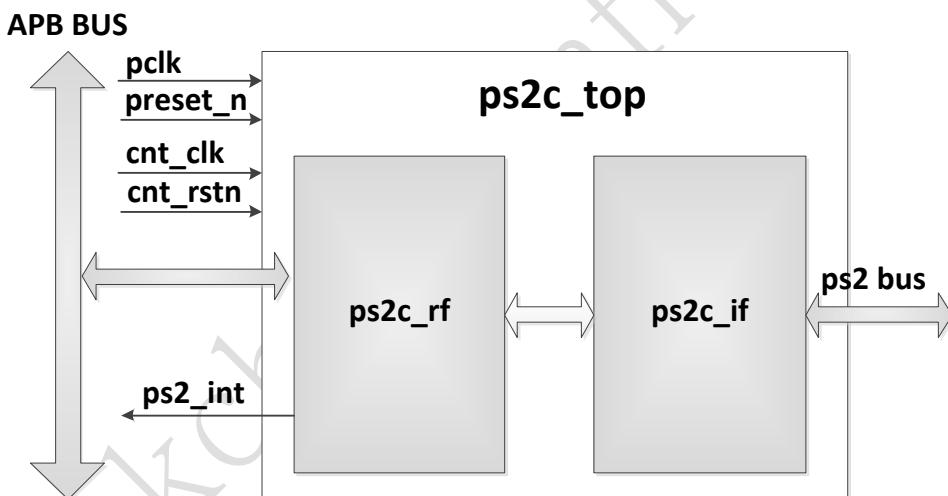


Fig. 43-1 PS/2 controller architecture

#### PS2C\_RF

PS2C\_RF module is used to control the PS/2 controller operation by the host with APB interface. It implements the register configuration and the interrupt functionality. It also collects the status signals to be queried by the host.

#### PS2C\_IF

PS2C\_IF module implements the PS/2 master operation for transmitting data to and receiving data from other PS/2 devices. The PS/2 master operation works at the pclk domain.

PS2C\_IF module also implements timing requirement counters and all the counters work at the cnt\_clk domain.

#### PS2C\_TOP

PS2C\_TOP module is the top module of the PS/2 controller.

### 43.3 Function description

This chapter provides a description about the functions and behavior of PS/2 controller.

The PS/2 controller supports only Master function. The operations of PS/2 controller is divided to 3 parts and described separately: receiving mode, sending mode and inhibition mode.

#### 43.3.1 PS/2 receiving mode

Configure the PS2C\_CTRL[2:1] as 2'b00 and enable controller, the PS/2 controller will work at receiving mode. In receiving mode, the PS/2 controller can receive the data transmitted from the PS/2 device. The receiving timing is showed as fig,

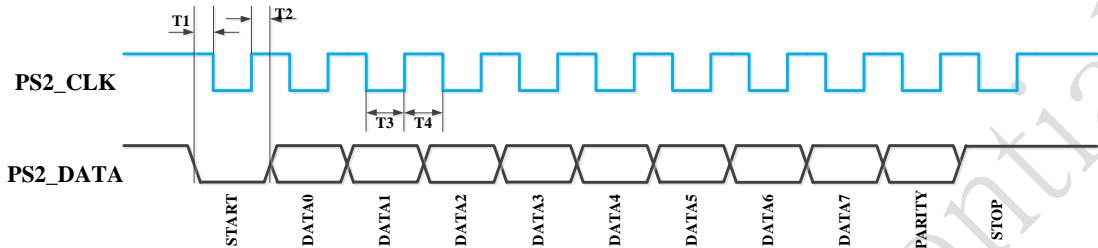


Fig. 43-2 PS/2 host receiving timing

- In receiving mode, the PS/2 controller waits to receive data transmitted from PS/2 device. Every transmission includes 11 bits data (one “start” bit, eight “information” bits, one “parity” bit and one “stop” bit).
- When negedge of ps2\_clk is captured and ps2\_data is low, the “start” bit is received. Then controller captures the following 10 bits data at the negedge of ps2\_clk.

After finish receiving a data package, the controller will generate the receiving finish status and receiving finish interrupt (if receiving finish interrupt is enabled). The host can get the receiving data by reading the PS2C\_RBR register. And if there is error in the data package, the controller will generate the error status.

In the controller, there is a receiving timeout dection circuit. This circuit is enabled by configuring the PS2C\_CTRL[3] as 1'b1. The receiving timeout period is set by configuring the PS2C\_RTR register. The unit of PS2C\_RTR is cnt\_clk cycle. If a data package receiving time exceeds the receiving timeout period after controller receives “start” bit, the controller will generate the receiving timeout status and receiving timeout interrupt (if receiving timeout interrupt is enabled).

#### 43.3.2 PS/2 sending mode

Configure the PS2C\_CTRL[2:1] as 2'b01 and enable controller, the PS/2 controller will work at sending mode. In sending mode, the PS/2 controller can send the data to the PS/2 device. The transmission timing is showed as fig,

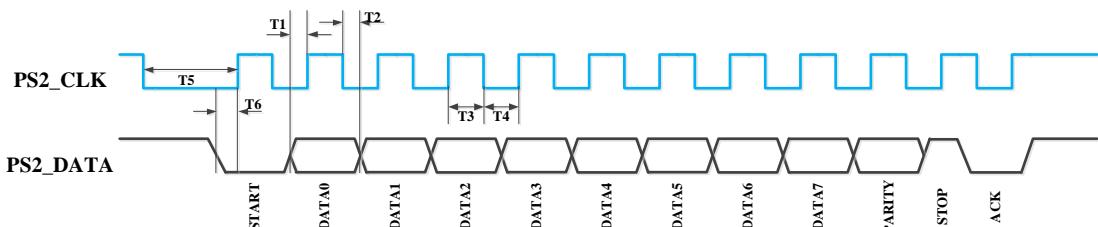


Fig. 43-3 PS/2 host sending timing

- In sending mode, the host sends data by writing PS2C\_TBR register which includes 8 information bits. Every transmission includes 12 bits data (one “start” bit, eight

"information" bits, one "parity" bit, one "stop" bit and one "ack" bit from PS/2 device).

- When there is data to be sent to PS/2 device, controller pulls low the ps2\_clk for T5 time duration (at least 100 microseconds) to request the PS/2 device receiving data. During T5 time duration, controller needs to pull low the ps2\_data to low for T6 time duration (at least 5 microseconds). After T5 time duration, the "start" bit will be sent.
- After the "start" bit is sent, controller sends the following 10 bits data captured by PS/2 device at the posedge of ps2\_clk. The controller changes the sending bit data at the low duration of ps2\_clk, and T1 (at least 5 microseconds) and T2 (at least 5 microseconds) time duration must be met.
- When finish sending the 11 bits data, controller will capture the "ack" bit at the next negedge of ps2\_clk. If the "ack" bit is 1'b0, it represents data package sending success, otherwise represents sending error.

Host can configure the PS2C\_TRR1, PS2C\_TRR2 and PS2C\_TRR3 registers respectively to meet the time of T5, T6 and T2. The unit of PS2C\_TRR1, PS2C\_TRR2 and PS2C\_TRR3 is cnt\_clk cycle.

After finish sending a data package, the controller will generate the sending finish status and sending finish interrupt (if sending finish interrupt is enabled). And if the "ack" bit is 1'b1, the controller will generate the error status.

In the controller, there is a sending timeout dection circuit. This circuit is enabled by configuring the PS2C\_CTRL[4] as 1'b1. The sending timeout period is set by configuring the PS2C\_WTR register. The unit of PS2C\_WTR is cnt\_clk cycle. If a data package sending time exceeds the sending timeout period after controller sends request to device, the controller will generate the sending timeout status and sending timeout interrupt (if sending timeout interrupt is enabled).

### 43.3.3 PS/2 inhibition mode

Configure the PS2C\_CTRL[2:1] as 2'b10 and enable controller, the PS/2 controller will work at inhibition mode. In inhibition mode, the PS/2 controller will pull ps2\_clk low to inhibit the PS/2 device sending data. The controller will not release the inhibition until disable controller.

## 43.4 Register Description

This section describes the control/status registers of the design.

### 43.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
PS2C_CTRL	0x0000	W	0x00000000	PS2C control register
PS2C_RBR	0x0004	W	0x00000000	PS2C receiving buffer register
PS2C_TBR	0x0008	W	0x00000000	PS2C sending buffer register
PS2C_STAT	0x000c	W	0x00000010	PS2C status register
PS2C_IER	0x0010	W	0x00000000	PS2C interrupt enable register
PS2C_ICR	0x0014	W	0x00000000	PS2C interrupt clear register
PS2C_ISR	0x0018	W	0x00000000	PS2C interrupt status register
PS2C_TRR1	0x001c	W	0x00000b39	PS2C time require register1

Name	Offset	Size	Reset Value	Description
PS2C_TRR2	0x0020	W	0x000000ef	PS2C time require register2
PS2C_TRR3	0x0024	W	0x000000ef	PS2C time require register3
PS2C_RTR	0x0028	W	0x00000000	PS2C receiving timeout require register
PS2C_WTR	0x002c	W	0x00000000	PS2C sending timeout require register
PS2C_FLT	0x0030	W	0x00000000	PS2C filter width selection register

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

#### 43.4.2 Detail Register Description

##### PS2C\_CTRL

Address: Operational Base + offset (0x0000)

PS2C control register

Bit	Attr	Reset Value	Description
31:21	RO	0x0	reserved
20:16	WO	0x00	write_enable bit0~4 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software. ..... When bit 20=1, bit 4 can be written by software. When bit 20=0, bit 4 cannot be written by software.
15:5	RO	0x0	reserved
4	RW	0x0	ps2c_tm_timeout_en ps2c sending timeout enable This bit is used to enable the sending timeout circuit. 1'b0: ps2c sending timeout disable. 1'b1: ps2c sending timeout enable. Only when this bit is set, the PS2C_STAT[6](ps2c_tm_timeout) is access.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RW	0x0	ps2c_rv_timeout_en ps2c receiving timeout enable This bit is used to enable the receiving timeout circuit. 1'b0: ps2c receiving timeout disable. 1'b1: ps2c receiving timeout enable. Only when this bit is set, the PS2C_STAT[2](ps2c_rv_timeout) is access.
2:1	RW	0x0	ps2c_mode ps2c work mode selection 2'b00: ps2c works as receiving mode. 2'b01: ps2c works as sending mode. 2'b10: ps2c works as inhibition mode, inhibits the ps2 device send data. 2'b11: reserved.
0	RW	0x0	ps2c_enable ps2c enable signal 1'b0: ps2c disable. 1'b1: ps2c enable.

**PS2C\_RBR**

Address: Operational Base + offset (0x0004)

PS2C receiving buffer register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RO	0x00	ps2c_rbuf ps2c receiving buffer

**PS2C\_TBR**

Address: Operational Base + offset (0x0008)

PS2C sending buffer register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	WO	0x00	ps2c_tbuf ps2c sending buffer

**PS2C\_STAT**

Address: Operational Base + offset (0x000c)

PS2C status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:23	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
22:21	WO	0x0	<p>write_enable2</p> <p>bit5~6 write enable</p> <p>When bit 21=1, bit 5 can be written by software.</p> <p>When bit 21=0, bit 5 cannot be written by software.</p> <p>When bit 22=1, bit 6 can be written by software.</p> <p>When bit 22=0, bit 6 cannot be written by software.</p>
20:19	RO	0x0	reserved
18:17	WO	0x0	<p>write_enable1</p> <p>bit1~2 write enable</p> <p>When bit 17=1, bit 1 can be written by software.</p> <p>When bit 17=0, bit 1 cannot be written by software.</p> <p>When bit 18=1, bit 2 can be written by software.</p> <p>When bit 18=0, bit 2 cannot be written by software.</p>
16:8	RO	0x0	reserved
7	RO	0x0	<p>ps2c_tm_busy</p> <p>ps2c sending busy</p> <p>This bit is set when write PS2C_TBR to start data sending to ps2 device, and is cleared when ps2c receives ack signal or happen sending timeout.</p> <p>1'b0: ps2c is in sending idle status.</p> <p>1'b1: ps2c is in sending busy status.</p>
6	RW	0x0	<p>ps2c_tm_timeout</p> <p>ps2c sending timeout</p> <p>When ps2c sending busy status exceeds the ps2c_tm_timeout_req_cnt_clk cycles time, this bit is set.</p> <p>1'b0: ps2c is not sending timeout.</p> <p>1'b1: ps2c is sending timeout.</p> <p>This bit is cleared by writing 1'b0.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	<p>ps2c_tm_err ps2c sending error When ps2c_tbe is 1'b1, this bit indicates whether the data sent to ps2 device is successfully received. 1'b0: ps2 device ack 1'b0 to ps2c. 1'b1: ps2 device ack 1'b1 to ps2c. This bit is cleared by writing 1'b0.</p>
4	RO	0x1	<p>ps2c_tbe ps2c sending buffer empty 1'b0: ps2c sending buffer not empty. 1'b1: ps2c sending buffer empty. This bit is cleared by writing the PS2C_TBR.</p>
3	RO	0x0	<p>ps2c_rv_busy ps2c receiving busy This bit is set when ps2c receives start signal, and is cleared when ps2c receives stop signal or happen receiving timeout. 1'b0: ps2c is in receiving idle status. 1'b1: ps2c is in receiving busy status.</p>
2	RW	0x0	<p>ps2c_rv_timeout ps2c receiving timeout When ps2c receiving busy status exceeds the ps2c_rv_timeout_req cnt_clk cycles time, this bit is set. 1'b0: ps2c is not receiving timeout. 1'b1: ps2c is receiving timeout. This bit is cleared by writing 1'b0.</p>
1	RW	0x0	<p>ps2c_rv_err ps2c receiving error When ps2c_ibf is 1'b1, this bit indicates whether the receiving data is odd parity error. 1'b0: odd parity sucess. 1'b1: odd parity error. This bit is cleared by writing 1'b0.</p>
0	RO	0x0	<p>ps2c_rbf ps2c receiving buffer full 1'b0: ps2c receiving buffer not full. 1'b1: ps2c receiving buffer full. This bit is cleared by reading the PS2C_RBR.</p>

**PS2C\_IER**

Address: Operational Base + offset (0x0010)

PS2C interrupt enable register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved
19:16	WO	0x0	<p>write_enable bit0~3 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software.</p> <p>.....</p> <p>When bit 19=1, bit 3 can be written by software. When bit 19=0, bit 3 cannot be written by software.</p>
15:4	RO	0x0	reserved
3	RW	0x0	<p>ps2c_tm_timeout_inten enable ps2c sending timeout interrupt 1'b0: disable interrupt. 1'b1: enable interrupt.</p>
2	RW	0x0	<p>ps2c_fsh_tm_inten enable ps2c finish sending one byte interrupt 1'b0: disable interrupt. 1'b1: enable interrupt.</p>
1	RW	0x0	<p>ps2c_rv_timeout_inten enable ps2c receiving timeout interrupt 1'b0: disable interrupt. 1'b1: enable interrupt.</p>
0	RW	0x0	<p>ps2c_fsh_rv_inten enable ps2c finish receiving one byte interrupt 1'b0: disable interrupt. 1'b1: enable interrupt.</p>

**PS2C\_ICR**

Address: Operational Base + offset (0x0014)

PS2C interrupt clear register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:20	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
19:16	WO	0x0	<p>write_enable bit0~3 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software. ..... When bit 19=1, bit 3 can be written by software. When bit 19=0, bit 3 cannot be written by software.</p>
15:4	RO	0x0	reserved
3	W1C	0x0	<p>ps2c_tm_timeout_intclr clear ps2c sending timeout interrupt 1'b0: not clear interrupt. 1'b1: clear interrupt.</p>
2	W1C	0x0	<p>ps2c_fsh_tm_intclr clear ps2c finish sending one byte interrupt 1'b0: not clear interrupt. 1'b1: clear interrupt.</p>
1	W1C	0x0	<p>ps2c_rv_timeout_intclr clear ps2c receiving timeout interrupt 1'b0: not clear interrupt. 1'b1: clear interrupt.</p>
0	W1C	0x0	<p>ps2c_fsh_rv_intclr clear ps2c finish receiving one byte interrupt 1'b0: not clear interrupt. 1'b1: clear interrupt.</p>

**PS2C\_ISR**

Address: Operational Base + offset (0x0018)

PS2C interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3	RO	0x0	<p>ps2c_tm_timeout_intst ps2c sending timeout interrupt status 1'b0: interrupt is not valid. 1'b1: interrupt is valid.</p>
2	RO	0x0	<p>ps2c_fsh_tm_intst ps2c finish sending one byte interrupt status 1'b0: interrupt is not valid. 1'b1: interrupt is valid.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RO	0x0	ps2c_rv_timeout_intst ps2c receiving timeout interrupt status 1'b0: interrupt is not valid. 1'b1: interrupt is valid.
0	RO	0x0	ps2c_fsh_rv_intst ps2c finish receiving one bype interrupt status 1'b0: interrupt is not valid. 1'b1: interrupt is valid.

**PS2C\_TRR1**

Address: Operational Base + offset (0x001c)

PS2C time require register1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000b39	ps2c_time_req1 ps2c time require 1 When ps2c sends data to ps2 device, this value defines the width of the low ps2_clk_o to send transmission request to the ps2 device. The unit is one cnt_clk cycle.

**PS2C\_TRR2**

Address: Operational Base + offset (0x0020)

PS2C time require register2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x000000ef	ps2c_time_req2 ps2c time require 2 When ps2c sends data to ps2 device, this value defines the width between first ps2_data_o falling edge and first ps2_clk_o rising edge. The unit is one cnt_clk cycle.

**PS2C\_TRR3**

Address: Operational Base + offset (0x0024)

PS2C time require register3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x000000ef	ps2c_time_req3 ps2c time require 3 When ps2c sending data to ps2 device, this value defines data change time after ps2_clk_i falling edge. The unit is one cnt_clk cycle.

**PS2C\_RTR**

Address: Operational Base + offset (0x0028)

PS2C receiving timeout require register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	ps2c_rv_timeout_req ps2c receiving timeout require time The unit is one cnt_clk cycle.

**PS2C\_WTR**

Address: Operational Base + offset (0x002c)

PS2C sending timeout require register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	ps2c_tm_timeout_req ps2c sending timeout require time The unit is one cnt_clk cycle.

**PS2C\_FLT**

Address: Operational Base + offset (0x0030)

PS2C filter width selection register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RW	0x00	ps2c_clk_flt_sel ps2c filter width selection for ps2_clk The unit is one cnt_clk cycle.

**43.5 Interface description**

Table 43-1 PS/2 controller Interface Description

<b>Module pin</b>	<b>Direction</b>	<b>Pad name</b>	<b>IOMUX</b>
PS/2 controller 0 Interface			
ps2_clk	I/O	GPIO8_A[0]	GPIO8A_IOMUX[1:0] = 01
ps2_data	I/O	GPIO8_A[1]	GPIO8A_IOMUX[3:2] = 01

**n Notes**

The PS/2 controller operation flow charts below are to describe how the software configures and performs a PS/2 transmission through this PS/2 controller. Descriptions are divided into 3 sections: receiving data mode, sending data mode and inhibition mode.

Note: Before change the controller's working mode, software needs to disable the controller firstly.

- Receiving data mode

**43.6 Application**

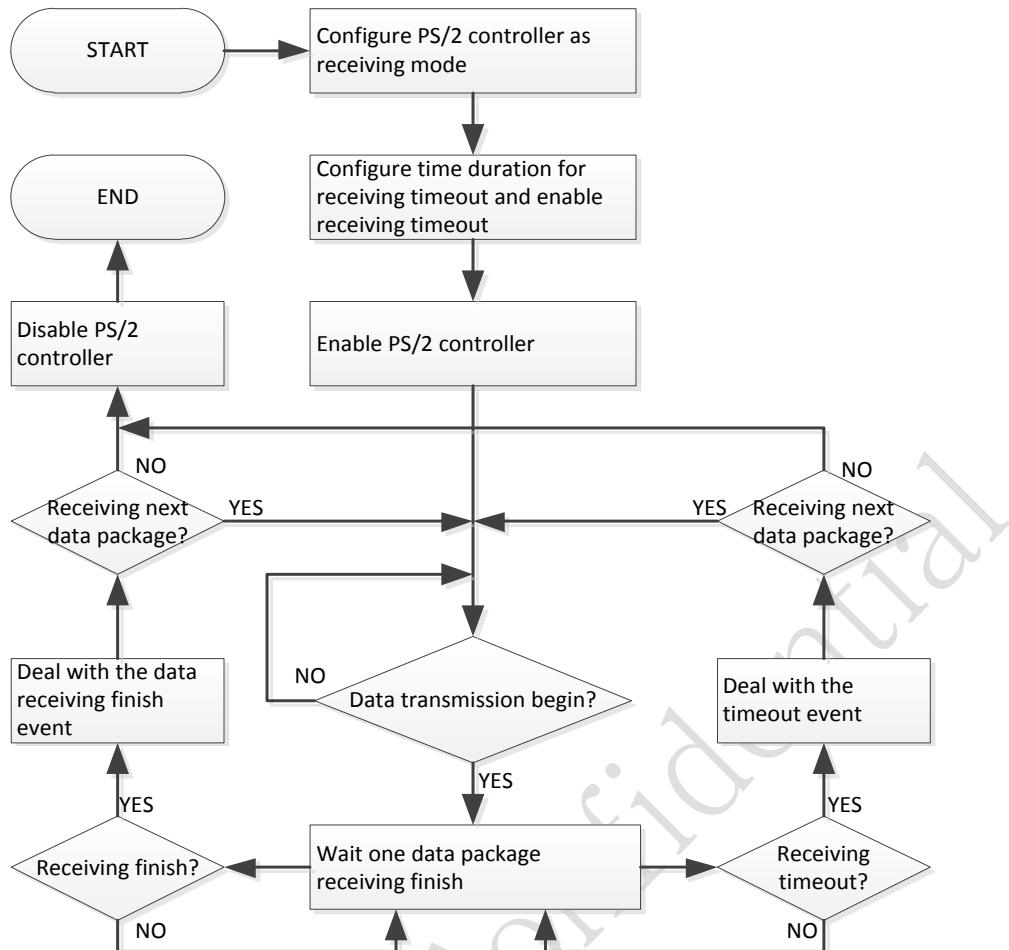


Fig. 43-4 Flow chat for PS/2 controller receiving data mode

- Sending data mode

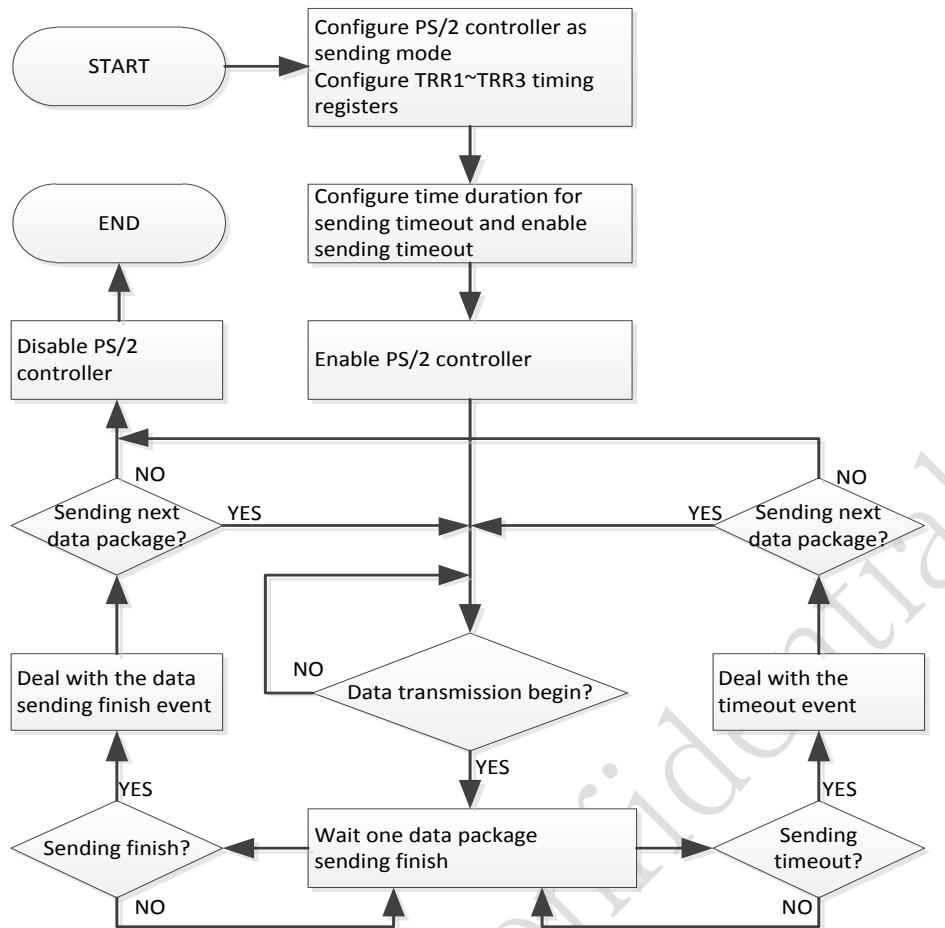


Fig. 43-5 Flow chat for PS/2 controller sending data mode

- Inhibition PS/2 device mode

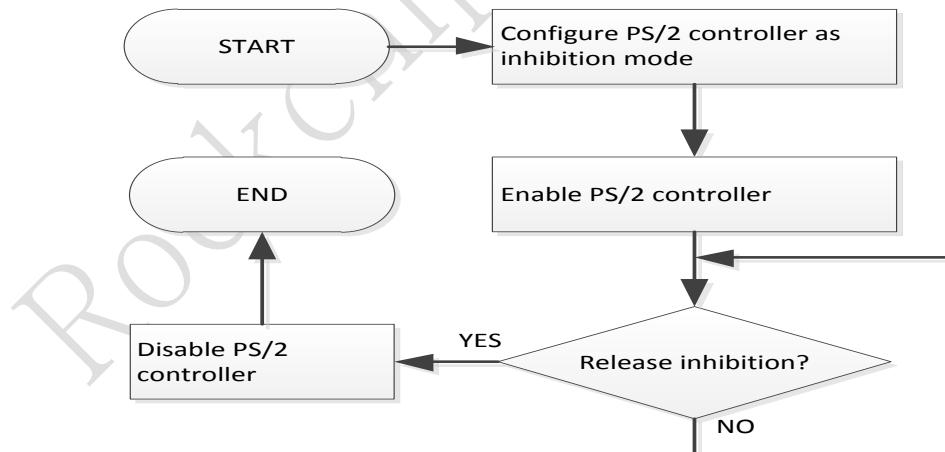


Fig. 43-6 Flow chat for PS/2 controller inhibition mode

## Chapter 44 Smart Card Controller

### 44.1 Overview

The Smart Card Reader (SCR) is a communication controller that transmits data between the superior system and the Smart Card. The controller can perform a complete smart card session, including card activation, card deactivation, cold/warm reset, Answer to Reset (ATR) response reception, data transfers, etc.

SCR supports the following features:

- Supports the ISO/IEC 7816-3:1997(E) and EMV2000 (4.0) specifications
- Performs functions needed for complete smart card sessions, including:
  - Card activation and deactivation
  - Cold/warm reset
  - Answer to Reset (ATR) response reception
  - Data transfers to and from the card
- Extensive interrupt support system
- Adjustable clock rate and bit (baud) rate
- Configurable automatic byte repetition
- Handles commonly used communication protocols:
  - T=0 for asynchronous half-duplex character transmission
  - T=1 for asynchronous half-duplex block transmission
- Automatic convention detection
- Configurable timing functions:
  - Smart card activation time
  - Smart card reset time
  - Guard time
  - Timeout timers
- Automatic operating voltage class selection
- Supports synchronous and any other non-ISO 7816 and non-EMV cards
- Advanced Peripheral Bus (APB) slave interface for easy integration with AMBA-based host systems

### 44.2 Block Diagram

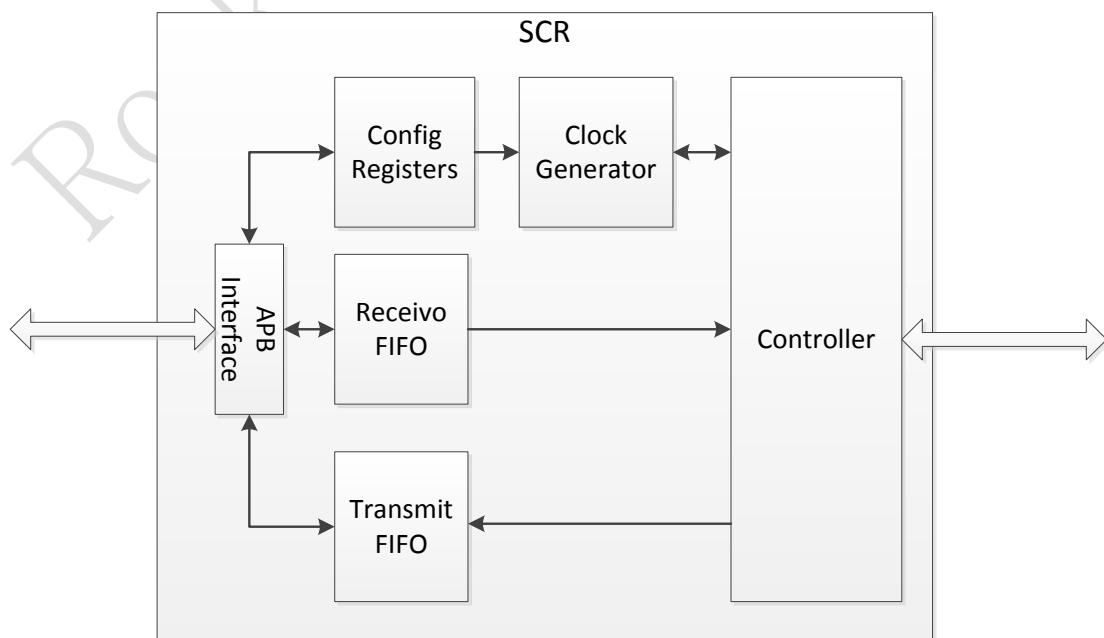


Fig. 44-1 SCR Block Diagram

The host processor gets access to PWM Register Block through the APB slave interface.

#### **44.2.1 APB Interface**

The host processor gets access to PWM Register Block through the APB slave interface.

#### **44.2.2 Configuration Registers**

The Configuration Registers block provides control over all functions of the Smart Card Reader

#### **44.2.3 Controller**

The Controller is the main block in the SCR core. This block controls receiving characters transmitted by the Smart Card, storing them in the RX FIFO, and transmitting them to the Smart Card. This block also performs card activation, deactivation, and cold and warm reset. After the card is reset, the Answer To Reset (ATR) sequence is received by the controller and stored in RX FIFO.

The parallel to serial conversion needed to transmit data from a Smart Card Reader to a Smart Card and the serial to parallel conversion needed to transmit data in the opposite direction is performed by the UART. The UART also performs the guard time, parity checking and character repeating functions.

#### **44.2.4 Receive FIFO**

The Receive FIFO is used to store the data received from the Smart Card until the data is read out by the superior system.

#### **44.2.5 Transmit FIFO**

The Transmit FIFO is used to store the data to be transmitted to the Smart Card.

#### **44.2.6 Clock Generator**

The Clock Generator generates the Smart Card Clock signal and the Baud Clock Impulse signal, used in timing the Smart Card Reader.)

### **44.3 Function Description**

A Smart Card session consists of following stages:

1. Smart Card insertion
2. Activation of contacts and cold reset sequence
3. Answer To Reset sequence (ATR)
4. Execution of transaction
5. Deactivation of contacts
6. Smart Card removal

#### **44.3.1 Smart Card Insertion**

A Smart Card session starts with the insertion of the Smart Card. This event is signaled to the SCR using the SCDETECT input. The SCPRESENT bit is set and also the SCINS interrupt is asserted (if enabled).

When the external card detect switch is not used, the input pin SCDETECT must be tied to inactive state.

### **44.3.2 Automatic operating voltage class selection**

There are three operating classes (1.8V - class C, 3V - class B and 5V - class A) defined in ISO/IEC 7816-3(2006) specification. Only 1.8V and 3.3V are supported by the SCR.

Before the activation of contacts, operating classes have to be enabled via bits VCC18, VCC33 in CTRL2 register. In case that no operating class is enabled, the controller performs activation for all two voltage classes (1.8V, 3V) in sequence.

When Smart Card Reader performs activation of contacts the lowest enabled voltage class is automatically applied first. When the first character start bit of ATR sequence is received, the selected voltage class is correct (even if the ATR is then received with errors). When the ATR sequence reception does not start, ATRFAIL interrupt is not activated, deactivation is performed and next higher enabled voltage class is applied. If the ATR sequence reception does not start and no other higher class is enabled was already applied the ATRFAIL interrupt is activated and the last applied voltage class remains active.

After the automatic voltage class selection is finished the selected class can be read from bits VCC18, VCC33 in CTRL2 register. If the automatic voltage class selection fails, these bits remain untouched.

There is a delay applied between deactivation of contacts with lower voltage class and activation of contacts with higher voltage class. This delay should be at least 10 ms according to the ISO/IEC 7816-3 specification.

### **44.3.3 Activation of Contacts and Cold Reset Sequence**

When the Smart Card is properly inserted and the ACT bit in CTRL2 register is asserted, the activation of contacts can be started. The duration of each part of the activation is the time  $T_a$ , which is equal to the ADEATIME register value. If no  $V_{pp}$  is necessary, the activation and deactivation part of  $V_{pp}$  can be omitted by clearing the AUTOADEAVPP bit in SCPADS register.

The Cold Reset sequence follows immediately after the activation. Time ( $T_c$ ) is the duration of the Reset. The EMV specification recommends that this value should be between 40000 and 45000. The activation of contacts and cold reset sequence is shown in Fig. 44-2.

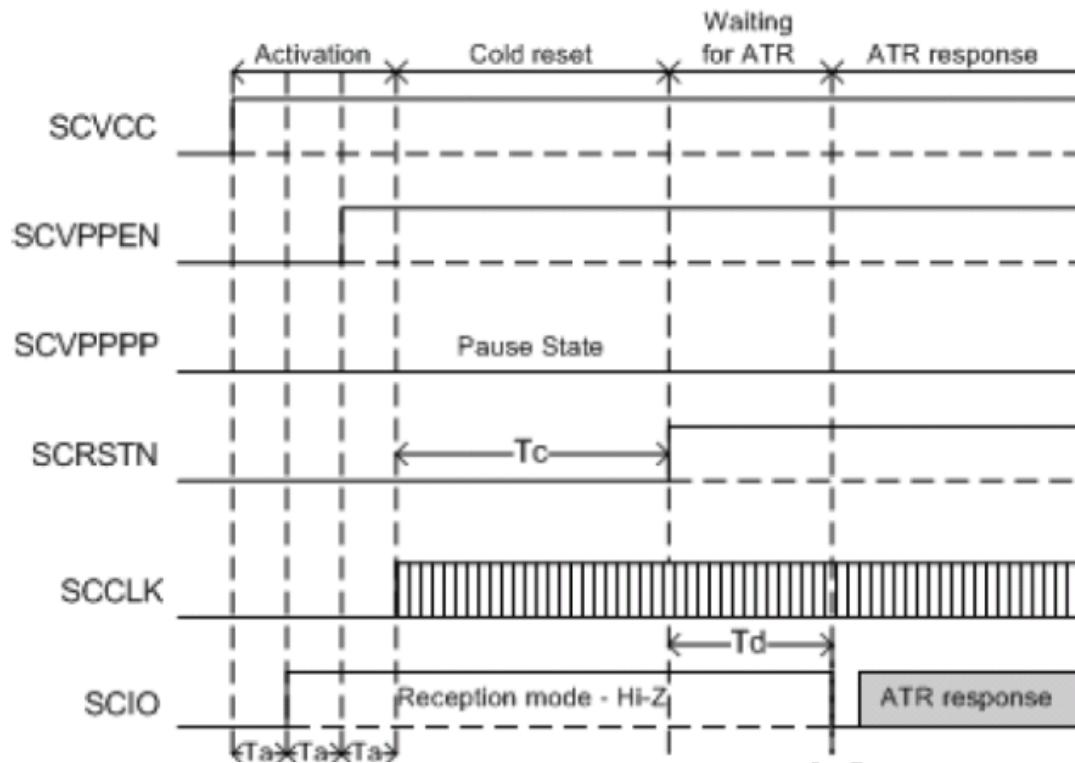


Fig. 44-2 Activation, Cold Reset and ATR

#### 44.3.4 Execution of Transaction

All transfers between the Smart Card Reader and a Smart Card are under the control of the superior system. It controls the number of characters sent to the Smart Card and it knows the number of characters expected to be returned from the Smart Card.

#### 44.3.5 Warm Reset

The Warm Reset sequence is initialized by setting the WRST bit in the CTRL2 register to '1'. Smart Card Reader drives the SCRSTN signal to '0' to perform the Warm Reset as shown in Fig. 44-3. After the SCRSTN assertion, the Warm Reset sequence then continues the same way as the Cold Reset sequence.

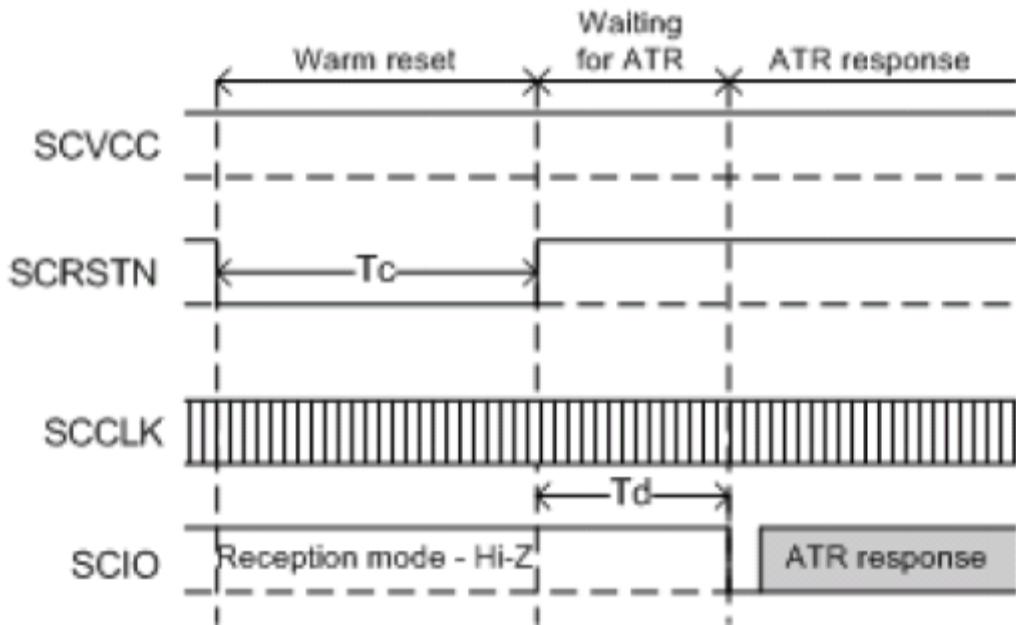


Fig. 44-3 Warm Reset and ATR

#### 44.3.6 Deactivation of Contacts

After the smart card reader detects the removal of the smart card (SCREM interrupt) or the superior system initiates deactivation by setting the DEACT bit in the CTRL2 register to '1', the deactivation is performed immediately as shown in . The duration time ( $T_a$ ), of each part of the deactivation sequence time is defined in the ADEATIME register.

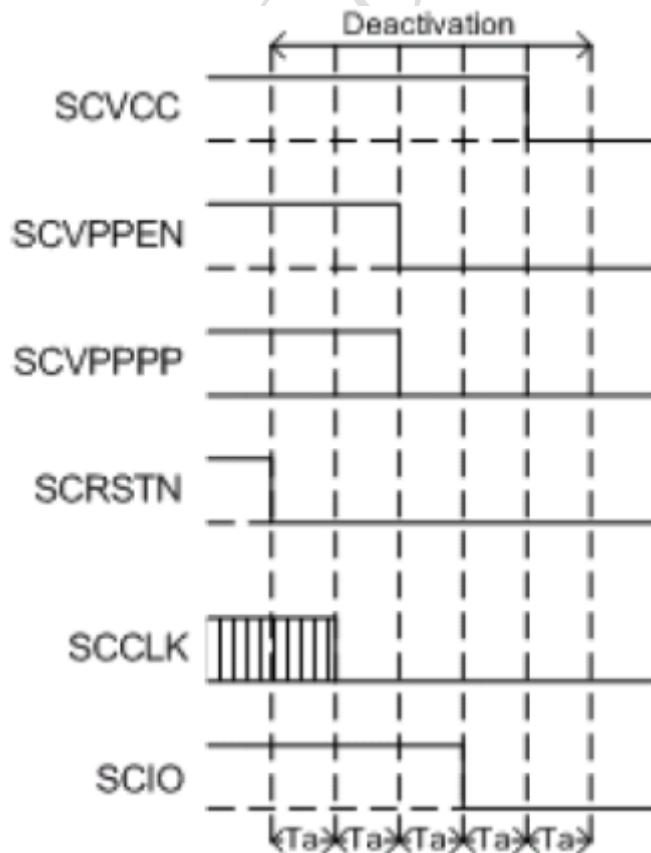


Fig. 44-4 Deactivation Sequence

## 44.4 Register description

### 44.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
SCR_CTRL1	0x0000	HW	0x0000	Control Register 1
SCR_CTRL2	0x0004	HW	0x0000	Control Register 2
SCR_SCPADS	0x0008	HW	0x0000	Smart Card Pads Register
SCR_INTEN1	0x000c	HW	0x0000	Interrupt Enable Register 1
SCR_INTSTAT1	0x0010	HW	0x0000	Interrupt Status Register 1
SCR_FIFOCTRL	0x0014	HW	0x0000	FIFO Control Register
SCR_LEGTXFICNT	0x0018	B	0x00	Legacy TX FIFO Counter
SCR_LEGRXFICNT	0x0019	B	0x00	Legacy RX FIFO Counter
SCR_RXFITH	0x001c	HW	0x0000	RX FIFO Threshold
SCR REP	0x0020	B	0x00	Repeat
SCR_SCCDDIV	0x0024	HW	0x0000	Smart Card Clock Divisor
SCR_BAUDDIV	0x0028	HW	0x0000	Baud Clock Divisor
SCR_SCGUTIME	0x002c	B	0x00	Smart Card Guardtime
SCR_ADEATIME	0x0030	HW	0x0000	Activation / Deactivation Time
SCR_LWRSTTIME	0x0034	HW	0x0000	Reset Duration
SCR_ATRSTARTLIMI T	0x0038	HW	0x0000	ATR Start Limit
SCR_C2CLIM	0x003c	HW	0x0000	Two Characters Delay Limit
SCR_INTEN2	0x0040	HW	0x0000	Interrupt Enable Register 2
SCR_INTSTAT2	0x0044	HW	0x0000	Interrupt Status Register 2
SCR_TXFITH	0x0048	HW	0x0000	TX FIFO Threshold
SCR_TXFIFOCNT	0x004c	HW	0x0000	TX FIFO Counter
SCR_RXFIFOCNT	0x0050	HW	0x0000	RX FIFO Counter
SCR_BAUTUNE	0x0054	B	0x00	Baud Tune Register
SCR_FIFO	0x0200	B	0x00	FIFO

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** -WORD (32 bits) access

### 44.4.2 Detail Register Description

#### SCR\_CTRL1

Address: Operational Base + offset (0x0000)

Control Register 1

Bit	Attr	Reset Value	Description
15	RW	0x0	GINTEN Global Interrupt Enable When high, INTERRUPT output assertion is enabled.
14	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
13	RW	0x0	TCKEN TCK enable When enabled all ATR bytes beginning from T0 are being XOR-ed. The result must be equal to TCK byte (when present). If the TCK byte does not match the computed value the ATR is considered to be malformed.
12	RW	0x0	ATRSTFLUSH ATR Start Flush FIFO When enabled, both FIFOs are flushed before the ATR is started.
11	RW	0x0	T0T1 T0/T1 Protocol Controls the using of T=0 or T=1 protocol. No character repeating is used when T=1 protocol is selected. The Character Guardtime (minimum delay between the leading edges of two consecutive characters) is reduced to 11 ETU when T=1 protocol is used and Guardtime value N = 255. The delay between the leading edge of the last received character and the leading edge of the first character transmitted is 16 ETU when T=0 protocol is used and 22 ETU when T=1 protocol is used.
10	RW	0x0	TS2FIFO TS to FIFO Enables to store the first ATR character TS in RX FIFO. During ideal card session there is no necessity to store TS character, so it can be disabled
9	RW	0x0	RXEN Receiving enable When enabled the characters sent by the Smart Card are received by the UART and stored in RX FIFO. Receiving is internally disabled while a transmission is in progress.
8	RW	0x0	TXEN Transmission enable When enabled the characters are read from TX FIFO and transmitted through UART to the Smart Card

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	CLKSTOPVAL Clock Stop Value The value of the scclk output during the clock stop state.
6	RW	0x0	CLKSTOP Clock Stop Clock Stop. When this bit is asserted and the smart card I/O line is in 'Z' state, the SCR core stops driving of the smart card clock signal after the CLKSTOPDELAY time expires. The smart card clock is restarted immediately after the CLKSTOP signal is deasserted. New character transmission can be started by superior system after the CLKSTARTDELAY time expires. The expiration of both times is signaled by the CLKSTOPRUN bit in the Interrupt registers. Reading '1' from this bit signals that the clock is stopped or CLKSTARTDELAY time not expired yet. Reading '0' from this bit signals that the clock is not stopped.
5:3	RO	0x0	reserved
2	RW	0x0	PECH2FIFO Character With Wrong Parity to FIFO Enables storage of the characters received with wrong parity in RX FIFO.
1	RW	0x0	INVORD Inverse Bit Ordering When High, inverse bit ordering convention(MSB-LSB) is used.
0	RW	0x0	INVLEV Inverse Bit Level When high, inverse level convention is used(A= '1', Z='0');

**SCR\_CTRL2**

Address: Operational Base + offset (0x0004)

Control Register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:8	RO	0x00	Reserved3 Reserved Reserved bits are hard-wired to zero

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	VCC50 Control 5V Smart Card Vcc Control 5V Smart Card Vcc. Setting of this bit allows selection of 5V Vcc for Smart Card session (Class A). After the selection of operating class is completed, this bit is in '1'.if this class was selected. Default value after reset is '0'..
6	RW	0x0	VCC33 Control 3V Smart Card Vcc Setting of this bit allows selection of 3V Vcc for Smart Card session (Class B). After the selection of operating class is completed, this bit is in '1'.if this class was selected. Default value after reset is '0'.
5	RW	0x0	VCC18 Control 1.8V Smart Card Vcc Control 1.8V Smart Card Vcc. Setting of this bit allows selection of 1.8V Vcc for Smart Card session (Class C). After the selection of operating class is completed, this bit is in '1'.if this class was selected. Default value after reset is '0'..
4	RW	0x0	DEACT Deactivation Setting of this bit initializes the deactivation sequence. When the deactivation is finished, the DEACT bit is automatically cleared.
3	RW	0x0	ACT Activation Setting of this bit initializes the activation sequence. When the activation is finished, the ACT bit is automatically cleared.
2	WO	0x0	WARMRST Warm Reset Command Writing '1'.to this bit initializes Warm Reset of the Smart Card. This bit is always read as '0'..
1:0	RO	0x0	reserved

**SCR\_SCPADS**

Address: Operational Base + offset (0x0008)

Smart Card Pads Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:10	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RO	0x0	SCPPRESENT Smart Card presented This bit is set to '1'.when the SCDETECT input is active at least for SCDETECTTIME
8	RW	0x0	DSCFCB Direct Smart Card Function Code Bit It provides direct access to SCFCB output
7	RW	0x0	DSCVPPPP Direct Smart Card Vpp Pause/Prog It provides direct access to SCVPPPP output
6	RW	0x0	DSCVPSEN Direct Smart Card Vpp Enable It provides direct access to SCVPSEN output
5	RW	0x0	AUTOADEAVPP Automatic Vpp Handling. When high, it enables automatic handling of DSCVPSEN and DSCVPPPP signals during activation and deactivation sequence.
4	RW	0x0	DSCVCC Direct Smart Card Vcc Direct Smart Card Vcc. When DIRACCPADS = '1'., the DSCVCC bit provides direct access to SCVCCx outputs. The appropriate SCVCC18, SCVCC33 and SCVCC50 outputs are driven according to state of bits VCC18, VCC33 and VCC50 in CTRL2 register.
3	RW	0x0	DSCRST Direct Smart Card Reset When DIRACCPADS = '1'., the DSCRST bit provides direct access to SCRST output
2	RW	0x0	DSCCLK Direct Smart Card Clock When DIRACCPADS = '1'., the DSCCLK bit provides direct access to SCCLK output
1	RW	0x0	DSCIO Direct Smart Card Input/Output When DIRACCPADS = '1'., the DSCIO bit provides direct access to SCIO pad.
0	RW	0x0	DIRACCPADS Direct Access To Smart Card Pads When high, it disables a serial interface functionality and enables direct control of the smart card pads using following 4 bits.

**SCR\_INTEN1**

Address: Operational Base + offset (0x000c)

Interrupt Enable Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RW	0x0	SCDEACT Smart Card Deactivation Interrupt When enabled, this interrupt is asserted after the Smart Card deactivation sequence is complete.
14	RW	0x0	SCACT Smart Card Activation Interrupt. When enabled, this interrupt is asserted after the Smart Card activation sequence is complete.
13	RW	0x0	SCINS Smart Card Inserted Interrupt When enabled, this interrupt is asserted after the smart card insertion
12	RW	0x0	SCREM Smart Card Removed Interrupt. When enabled, this interrupt is asserted after the smart card removal.
11	RW	0x0	ATRDONE ATR Done Interrupt When enabled, this interrupt is asserted after the ATR sequence is successfully completed.
10	RW	0x0	ATRFAIL ATR Fail Interrupt When enabled, this interrupt is asserted if the ATR sequence fails.
9	RW	0x0	RXTHRESHOLD RX FIFO Threshold Interrupt When enabled, this interrupt is asserted if the number of bytes in RX FIFO is equal or exceeds the RX FIFO threshold.
8	RW	0x0	C2CFULL Two Consecutive Characters Limit Interrupt When enabled, this interrupt is asserted if the time between two consecutive characters, transmitted between the Smart Card and the Reader in both directions, is equal the Two Characters Delay Limit described below. The C2CFULL interrupt is internally enabled from the ATR start to the deactivation or ATR restart initialization. It is recommended to use this counter to detect unresponsive Smart Cards.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7	RW	0x0	RXPERR Reception Parity Error Interrupt When enabled, this interrupt is asserted after the character with wrong parity was received when the number of repeated receptions exceeds RXREPEAT value or T=1 protocol is used
6	RW	0x0	TXPERR Transmission Parity Error Interrupt. When enabled, this interrupt is asserted if the Smart Card signals wrong character parity during the guardtime after the character transmission was repeated TXREPEAT-times
5	RW	0x0	RXDONE Reception Done Interrupt When enabled, this interrupt is asserted after a character was received from the Smart Card.
4	RW	0x0	TXDONE Transmission Done Interrupt When enabled, this interrupt is asserted after one character was transmitted to the Smart Card.
3	RW	0x0	CLKSTOPRUN Smart Card Clock Stop Interrupt When enabled, this interrupt is asserted in two cases: 1. When the smart card clock is stopped (after CLOCKSTOP assertion). 2. When the new character transfer can be started (the smart card clock is fully running after CLOCKSTOP de-assertion).
2	RW	0x0	RXFIFULL RX FIFO Full Interrupt When enabled, this interrupt is asserted if the RX FIFO is filled up.
1	RW	0x0	TXFIEMPTY TX FIFO Empty Interrupt. When enabled, this interrupt is asserted if the TX FIFO is emptied out.
0	RW	0x0	TXFIDONE TX FIFO Done Interrupt When enabled, this interrupt is asserted after all bytes from TX FIFO were transferred to the Smart Card

**SCR\_INTSTAT1**

Address: Operational Base + offset (0x0010)

Interrupt Status Register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15	RW	0x0	SCDEACT Smart Card Deactivation Interrupt When enabled, this interrupt is asserted after the Smart Card deactivation sequence is complete.
14	RW	0x0	SCACT Smart Card Activation Interrupt. When enabled, this interrupt is asserted after the Smart Card activation sequence is complete.
13	RW	0x0	SCINS Smart Card Inserted Interrupt When enabled, this interrupt is asserted after the smart card insertion
12	RW	0x0	SCREM Smart Card Removed Interrupt. When enabled, this interrupt is asserted after the smart card removal.
11	RW	0x0	ATRDONE ATR Done Interrupt When enabled, this interrupt is asserted after the ATR sequence is successfully completed.
10	RW	0x0	ATRFAIL ATR Fail Interrupt When enabled, this interrupt is asserted if the ATR sequence fails.
9	RW	0x0	RXTHRESHOLD RX FIFO Threshold Interrupt When enabled, this interrupt is asserted if the number of bytes in RX FIFO is equal or exceeds the RX FIFO threshold.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	RW	0x0	C2CFULL Two Consecutive Characters Limit Interrupt When enabled, this interrupt is asserted if the time between two consecutive characters, transmitted between the Smart Card and the Reader in both directions, is equal the Two Characters Delay Limit described below. The C2CFULL interrupt is internally enabled from the ATR start to the deactivation or ATR restart initialization. It is recommended to use this counter to detect unresponsive Smart Cards.
7	RW	0x0	RXPERR Reception Parity Error Interrupt When enabled, this interrupt is asserted after the character with wrong parity was received when the number of repeated receptions exceeds RXREPEAT value or T=1 protocol is used
6	RW	0x0	TXPERR Transmission Parity Error Interrupt. When enabled, this interrupt is asserted if the Smart Card signals wrong character parity during the guardtime after the character transmission was repeated TXREPEAT-times
5	RW	0x0	RXDONE Reception Done Interrupt When enabled, this interrupt is asserted after a character was received from the Smart Card.
4	RW	0x0	TXDONE Transmission Done Interrupt When enabled, this interrupt is asserted after one character was transmitted to the Smart Card.
3	RW	0x0	CLKSTOPRUN Smart Card Clock Stop Interrupt When enabled, this interrupt is asserted in two cases: 1. When the smart card clock is stopped (after CLOCKSTOP assertion). 2. When the new character transfer can be started (the smart card clock is fully running after CLOCKSTOP de-assertion).

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x0	RXFIFULL RX FIFO Full Interrupt When enabled, this interrupt is asserted if the RX FIFO is filled up.
1	RW	0x0	TXFIEMPTY TX FIFO Empty Interrupt. When enabled, this interrupt is asserted if the TX FIFO is emptied out.
0	RW	0x0	TXFIDONE TX FIFO Done Interrupt When enabled, this interrupt is asserted after all bytes from TX FIFO were transferred to the Smart Card

**SCR\_FIFOCTRL**

Address: Operational Base + offset (0x0014)

FIFO Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:11	RO	0x0	reserved
10	WO	0x0	RXFIFLUSH Flush RX FIFO RX FIFO is flushed, when '1'.is written to this bit.
9	RO	0x0	RXFIFULL RX FIFO Full RX FIFO Full
8	RO	0x0	TXFIEMPTY RX FIFO Empty Field0000 Description
7:3	RO	0x0	reserved
2	WO	0x0	TXFIFLUSH Flush TX FIFO. TX FIFO is flushed, when '1'.is written to this bit.
1	RO	0x0	TXFIFULL TX FIFO Full TX FIFO Full
0	RO	0x0	TXFIEMPTY TX FIFO Empty. TX FIFO Empty.

**SCR\_LEGTXFICNT**

Address: Operational Base + offset (0x0018)

Legacy TX FIFO Counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	RO	0x00	LEGTXFICNT Legacy TX FIFO Counter It is equal to TX FIFO Counter up to value 255. All values above 255 are read as 255. It is recommended to use the 16-bit TX FIFO Counter instead of this register.

**SCR\_LEG\_RXFICNT**

Address: Operational Base + offset (0x0019)

Legacy RX FIFO Counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	RO	0x00	LEG_RXFICNT Legacy RX FIFO Counter It is equal to RX FIFO Counter up to value 255. All values above 255 are read as 255. It is recommended to use the 16-bit RX FIFO Counter instead of this register.

**SCR\_RXFITH**

Address: Operational Base + offset (0x001c)

RX FIFO Threshold

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	RXFITH RX FIFO Threshold The interrupt is asserted when the number of bytes it receives is equal to, or exceeds the threshold

**SCR REP**

Address: Operational Base + offset (0x0020)

Repeat

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:4	RW	0x0	RXREP RX Repeat This is a 4-bit, read/write register that specifies the number of attempts to request character re-transmission after wrong parity was detected. The re-transmission of the character is requested using the 1 ETU long error signal during the guardtime

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	<p>TXREP TX Repeat</p> <p>This is a 4-bit, read/write register that specifies the number of attempts to re-transmit the character after the Smart Card signals the wrong parity during the guardtime.</p>

**SCR\_SCCDDIV**

Address: Operational Base + offset (0x0024)

Smart Card Clock Divisor

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	<p>SCCDDIV Smart Card Clock Divisor</p> <p>This is a 16-bit, read/write register that defines the divisor value used to generate the Smart Card Clock from the system clock.</p>

**SCR\_BAUDDIV**

Address: Operational Base + offset (0x0028)

Baud Clock Divisor

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	<p>BAUDDIV Baud Clock Divisor</p> <p>This is a 16-bit, read/write register that defines a divisor value used to generate the Baud Clock impulses from the system clock</p>

**SCR\_SCGUTIME**

Address: Operational Base + offset (0x002c)

Smart Card Guardtime

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	RW	0x00	<p>SCGUTI Smart Card Guardtime</p> <p>This is an 8-bit, read/write register that sets a delay at the end of each character transmitted from the Smart Card Reader to the Smart Card. The value is in Elementary Time Units (ETU). The parity error is besides signaled during the guardtime</p>

**SCR\_ADEATIME**

Address: Operational Base + offset (0x0030)

Activation / Deactivation Time

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:8	RW	0x00	ADEATIME Activation / Deactivation Time Sets the duration of each part of the activation and deactivation sequence. The value is in Smart Card Clock Cycles.
7:0	RW	0x00	Reserved Reserved Reserved bits are hard-wired to zero.

**SCR\_LOWRSTTIME**

Address: Operational Base + offset (0x0034)

Reset Duration

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:8	RW	0x00	LOWRSTTIME Reset Duration Sets the duration of the smart card reset sequence. This value is same for the cold and warm reset. The value is in terms of smart card clock cycles.
7:0	RW	0x00	Reserved Reserved Bits (7:0) of this register are hard-wired to zero.

**SCR\_ATRSTARTLIMIT**

Address: Operational Base + offset (0x0038)

ATR Start Limit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:8	RW	0x00	ATRSTARTLIMIT ATR Start Limit Defines the maximum time between the rising edge of the SCRSTN signal and the start of ATR response. The value is in terms of smart card clock cycles
7:0	RW	0x00	Reserved Reserved Bits (7:0) of this register are hard-wired to zero

**SCR\_C2CLIM**

Address: Operational Base + offset (0x003c)

Two Characters Delay Limit

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	C2CLIM Two Characters Delay Limit This is a 16-bit, read/write register that sets the maximum time between the leading edges of two, consecutive characters. The value is in 16 ETUs.

**SCR\_INTEN2**

Address: Operational Base + offset (0x0040)

Interrupt Enable Register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:2	RO	0x0	reserved
1	RW	0x0	TCKERR TCK Error Interrupt. When enabled, this interrupt is asserted if the TCK byte does not match computed value.
0	RW	0x0	TXTHRESHOLD TX FIFO Threshold Interrupt When enabled, this interrupt is asserted if the number of bytes in TX FIFO is equal or less than the TX FIFO threshold.

**SCR\_INTSTAT2**

Address: Operational Base + offset (0x0044)

Interrupt Status Register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:2	RO	0x0	reserved
1	RW	0x0	TCKERR TCK Error Interrupt When enabled, this interrupt is asserted if the TCK byte does not match computed value.
0	RW	0x0	TXTHRESHOLD TX FIFO Threshold Interrupt When enabled, this interrupt is asserted if the number of bytes in TX FIFO is equal or less than the TX FIFO threshold.

**SCR\_TXFITH**

Address: Operational Base + offset (0x0048)

TX FIFO Threshold

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RW	0x0000	TXFITH TX FIFO Threshold The interrupt is asserted when the number of bytes in TX FIFO is equal or less than the threshold

**SCR\_TXFIFOCNT**

Address: Operational Base + offset (0x004c)

TX FIFO Counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RO	0x0000	TXFIFOCNT TX FIFO Counter This is a 16-bit, read-only register that provides the number of bytes stored in the RX FIFO

**SCR\_RXFIFOCNT**

Address: Operational Base + offset (0x0050)

RX FIFO Counter

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
15:0	RO	0x0000	RXFIFOCNT RX FIFO Counter This is a 16-bit, read-only register that provides the number of bytes stored in the RX FIFO.

**SCR\_BAUTUNE**

Address: Operational Base + offset (0x0054)

Baud Tune Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:4	RO	0x0	reserved
3:0	RW	0x0	BAUTUNE Baud Tune Register This is a 3-bit, read/write register that defines an additional value used to increase the accuracy of the Baud Clock impulses

**SCR\_FIFO**

Address: Operational Base + offset (0x0200)

FIFO

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

Bit	Attr	Reset Value	Description
7:0	RW	0x00	FIFO FIFO This is an 8-bit, read/write register that provides access to the receive and transmit FIFO buffers. The TX FIFO is accessed during the APB write transfer. The RX FIFO is accessed during the APB read transfer. All read/write accesses at address range 200h-3ffh are redirected to the FIFO.

## 44.5 Interface Description

Table 44-1 IOMUX Setting

Module Pin	IO	Pad Name	IOMUX Setting
scclk_0	O	I2C1SENSORscl_SCclk_GPIO1830gpio8a	GPIO8A_IOMUX[11:10]= 2'b10
scclk_1	O	SPI2txd_SCclk_GPIO1830gpio8b1	GPIO8B_IOMUX[3:2]= 2'b10
scrst_0	O	I2C1SENSORsda_SCrst_GPIO1830gpio8a4	GPIO8A_IOMUX[9:8]= 2'b10
scrst_1	O	SPI2rxr_SCrst_GPIO1830gpio8b0	GPIO8B_IOMUX[1:0]= 2'b10
scvcc18v	O	PS2clk_SCvcc18v_GPIO1830gpio8a0	GPIO8A_IOMUX[1:0]= 2'b10
scvcc33v	O	PS2data_SCvcc33v_GPIO1830gpio8a1	GPIO8A_IOMUX[3:2]= 2'b10
scdetect_0	O	SPI2csn0_SCdetect_GPIO1830gpio8a7	GPIO8A_IOMUX[15:14]= 2'b10
scdetect_1	O	SCdetectt1_GPIO1830gpio8a2	GPIO8A_IOMUX[5:4]= 2'b01
sci0_0	I/O	SPI2clk_SCio_GPIO1830gpio8a6	GPIO8A_IOMUX[13:12]= 2'b10
sci0_1	I/O	SPI2csn1_SCIot1_GPIO1830gpio8a3	GPIO8A_IOMUX[7:6]= 2'b10

Notes: 1. I=Input, O=Output, I/O=Input/Output, bidirectional

2. Pull up scio pin when data transaction

## 44.6 Application Notes

### 44.6.1 SCR Clock

The Smart Card Clock signal is used as the main clock for the smart card. Its frequency can be adjusted using the Smart Card Clock Divisor (SCCDIV). This value is used to divide the system clock.

The SCCLK frequency is given by the following equation:

$$\text{SCCLK}_{\text{freq}} = \frac{\text{CLK}_{\text{freq}}}{2 * (\text{SCCDIV} + 1)}, \quad \text{SCCDIV} \cong \frac{\text{CLK}_{\text{freq}}}{2 * \text{SCCLK}_{\text{freq}}} - 1$$

SCCLK\_freq – Smart Card Clock Frequency

### CLK\_freq- System Clock Frequency

The Baud Clock Impulse signal is used to transmit and receive serial data between the Smart Card Reader and the Smart Card. The baud rate can be modified using the Baud Clock Divisor (BAUDDIV) which is used to divide the system clock. The BAUDDIV value must be  $\geq 4$ . The BAUD rate is given by

the following equation:

$$\text{BAUD}_{\text{rate}} = \frac{\text{CLK\_freq}}{2 * (\text{BAUDDIV} + 1)}$$

The duration of one bit, Elementary Time Unit (ETU) and parameters F and D are defined in the ISO/IEC 7816-3 specification.

$$\frac{1}{\text{BAUD}_{\text{rate}}} \cong \text{ETU} = \frac{F}{D} * \frac{1}{\text{SCCLK}_{\text{freq}}} * \frac{F}{D} \cong \frac{\text{BAUDDIV} + 1}{\text{SCCDIV} + 1}$$

BAUDDIV equation based on SCCDIV value and Smart Card parameters F and D is following:

$$\text{BAUDDIV} \cong (\text{SCCDIV} + 1) * \frac{F}{D} - 1$$

During the first answer to reset response after the cold reset, the initial ETU must be equal to 372 Smart Card Clock Cycles (given by parameters F=372 and D=1). In this case, the BAUDDIV should be:

$$\text{BAUDDIV} \cong (\text{SCCDIV} + 1) * \frac{372}{1} - 1$$

After the ATR is completed, the BAUDDIV register value can be changed according to Smart Card parameters F and D.

Baud Tune Register (BAUDTUNE) 3-bit value that can be used to increase the accuracy of the Baud Clock impulses timing by using the BAUDTUNE Increment from Table listed below in combination with BAUDDIV register value.

Table 44-2 BAUDTUNE register

BAUDTUNE	000	001	010	011	100	101	110	111
BAUDTUNE <sub>INCR</sub>	+0	+0.125	0.25	+0.375	+0.5	+0.625	+0.75	+0.875

$$\text{BAUDDIV} + \text{BAUDTUNE}_{\text{INCR}} \cong (\text{SCCDIV} + 1) * \frac{F}{D} - 1$$

The BAUDDIV register value (nearest integer) can be computed using following equation:

$$\text{BAUDDIV} \cong (\text{SCCDIV} + 1) * \frac{F}{D} - 1 - \text{BAUDTUNE}_{\text{INCR}}$$

## Chapter 45 SAR-ADC

### 45.1 Overview

The ADC is a 3-channel signal-ended 10-bit Successive Approximation Register (SAR) A/D Converter. It uses the supply and ground as its reference which avoids the use of any external reference. It converts the analog input signal into 10-bit binary digital codes at maximum conversion rate of 100KSPS with 1MHz A/D converter clock.

### 45.2 Block Diagram

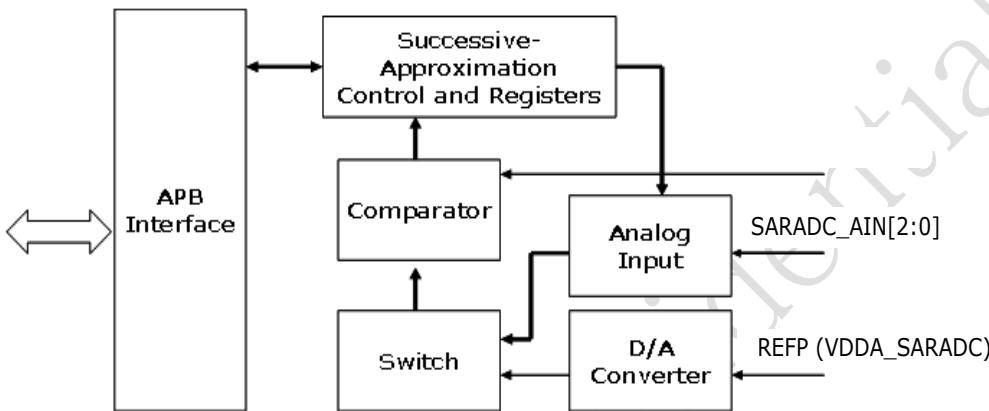


Fig. 45-1 RK3288 SAR-ADC block diagram

#### Successive-Approximate Register and Control Logic Block

This block is exploited to realize binary search algorithm, storing the intermediate result and generate control signal for analog block.

#### Comparator Block

This block compares the analog input SARADC\_AIN[2:0] with the voltage generated from D/A Converter, and output the comparison result to SAR and Control Logic Block for binary search. Three level amplifiers are employed in this comparator to provide enough gain.

### 45.3 Function description

In RK3288, SAR-ADC works at single-sample operation mode.

This mode is useful to sample an analog input when there is a gap between two samples to be converted. In this mode START is asserted only on the rising edge of CLKIN where conversion is needed. At the end of every conversion EOC signal is made high and valid output data is available at the rising edge of EOC. The detailed timing diagram will be shown in the following.

### 45.4 Register Description

This section describes the control/status registers of the design.

#### 45.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SARADC_DATA	0x0000	W	0x00000000	This register contains the data after A/D Conversion.
SARADC_STAS	0x0004	W	0x00000000	The status register of A/D Converter.
SARADC_CTRL	0x0008	W	0x00000000	The control register of A/D Converter.
SARADC_DLY_PU_SOC	0x000c	W	0x00000000	delay between power up and start command

Notes: Size: **B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**- WORD (32 bits) access

#### 45.4.2 Detail Register Description

##### SARADC\_DATA

Address: Operational Base + offset (0x0000)

This register contains the data after A/D Conversion.

Bit	Attr	Reset Value	Description
31:10	RO	0x0	reserved
9:0	RO	0x000	adc_data A/D value of the last conversion (DOUT[9:0]).

##### SARADC\_STAS

Address: Operational Base + offset (0x0004)

The status register of A/D Converter.

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	adc_status ADC status (EOC) 1'b0: ADC stop 1'b1: Conversion in progress

##### SARADC\_CTRL

Address: Operational Base + offset (0x0008)

The control register of A/D Converter.

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	int_status Interrupt status. This bit will be set to 1 when end of conversion. Set 0 to clear the interrupt.
5	RW	0x0	int_en Interrupt enable. 1'b0: Disable 1'b1: Enable
4	RO	0x0	reserved

Bit	Attr	Reset Value	Description
3	RW	0x0	adc_power_ctrl ADC power down control bit 1'b0: ADC power down 1'b1: ADC power up and reset start signal will be asserted (DLY_PU_SOC+2) sclk clock period later after power up
2:0	RW	0x0	adc_input_src_sel ADC input source selection (CH_SEL[2:0]). 3'b000: Input source 0 (SARADC_AIN[0]) 3'b001: Input source 1 (SARADC_AIN[1]) 3'b010: Input source 2 (SARADC_AIN[2]) Others : Reserved

**SARADC\_DLY\_PU\_SOC**

Address: Operational Base + offset (0x000c)

delay between power up and start command

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5:0	RW	0x08	DLY_PU_SOC delay between power up and start command The start signal will be asserted (DLY_PU_SOC + 2) sclk clock period later after power up

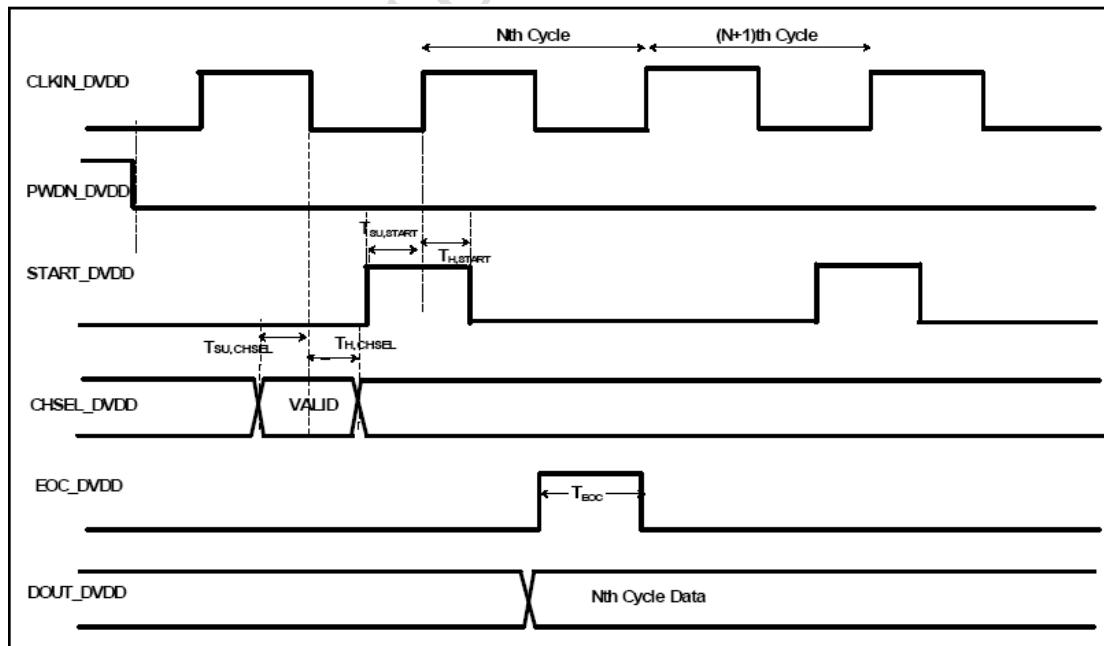
**45.5 Timing Diagram**

Fig. 45-2 SAR-ADC timing diagram in single-sample conversion mode

The following table has shows the detailed value for timing parameters in the above diagram.

Table 45-1 RK3288 SAR-ADC timing parameters list

Timing	Symbol	Value			Unit	Description
		Min	Typ	Max		
START_OF_CONV Setup time	TSU,START	5			ns	Set Up time for START_OF_CONV w.r.t CLKIN rising edge
START_OF_CONV Hold time	TH,START	5			ns	Hold time for START_OF_CONV w.r.t CLKIN rising edge
CHSEL setup time	TSU,CHSEL	5			ns	Set Up time for CHSEL w.r.t CLKIN falling edge
CHSEL Hold time	TH,CHSEL	5			ns	Hold time for CHSEL w.r.t CLKIN falling edge
Data Setup	TSU,DATA	400		900	ns	Set Up time for output data w.r.t either CLKIN rising edge or END_OF_CONV falling edge
Data Hold	TH,DATA	100		600	ns	Hold time for output data w.r.t either CLKIN rising edge or END_OF_CONV falling edge
Data access time	TDAC	100		600	ns	Valid data w.r.t CLKIN rising edge
Delay time	TDelay			5	ns	Delay between Valid data and EOC DVDD rising edge
EOC Pulse Width (max frequency)	TEOC	400		900	ns	Pulse width of EOC
CLKIN Rise Time	TCR			2	ns	CLKIN Rise Time
CLKIN Fall Time	TCF			2	ns	CLKIN Fall Time
CLK Pulse Width(Duty Cycle)	TCPW	45		55	%	CLKIN High/Low Time Period
CLK Period	TCP	1			us	CLKIN Time Period

## 45.6 Application Notes

Steps of adc conversion:

- Write SARADC\_CTRL[3] as 0 to power down adc converter.
- Write SARADC\_CTRL[2:0] as n to select adc channel(n).
- Write SARADC\_CTRL[5] as 1 to enable adc interrupt.
- Write SARADC\_CTRL[3] as 1 to power up adc converter.
- Wait for adc interrupt or poll SARADC\_STAS register to assert whether the coversion is completed
- Read the conversion result from SARADC\_DATA[9:0]

Note: The A/D converter was designed to operate at maximum 1MHZ.

## Chapter 46 Temperature-Sensor ADC(TS-ADC)

### 46.1 Overview

TS-ADC Controller module supports user-defined mode and automatic mode. User-defined mode refers, TSADC all the control signals entirely by software writing to register for direct control. Automatic mode refers to the module automatically poll TSADC output, and the results were checked. If you find that the temperature High in a period of time, an interrupt is generated to the processor down-measures taken; if the temperature over a period of time High, the resulting TSHUT gave CRU module, let it reset the entire chip, or via GPIO give PMIC.

TS-ADC Controller supports the following features:

- Support User-Defined Mode and Automatic Mode
- In User-Defined Mode, start\_of\_conversion can be controlled completely by software, and also can be generated by hardware.
- In Automatic Mode, the temperature of alarm interrupt can be configurable
- In Automatic Mode, the temperature of system reset can be configurable
- Support to 4 channel TS-ADC, the temperature criteria of each channel can be configurable
- In Automatic Mode, the time interval of temperature detection can be configurable
- In Automatic Mode, when detecting a high temperature, the time interval of temperature detection can be configurable
- High temperature debounce can be configurable

### 46.2 Block Diagram

TS-ADC controller comprises with:

- APB Interface
- TS-ADC control logic

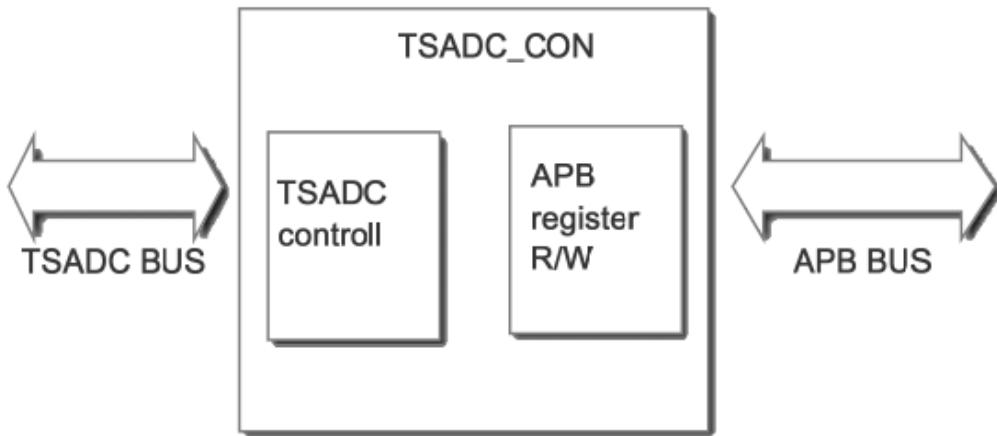


Fig. 46-1 TS-ADC Controller Block Diagram

### 46.3 Function Description

#### 46.3.1 APB Interface

There is an APB Slave interface in TS-ADC Controller, which is used to configure the TS-ADC Controller registers and look up the temperature from the temperature sensor.

### 46.3.2 TS-ADC Controller

This block is exploited to realize binary search algorithm, storing the intermediate result and generate control signal for analog block. This block compares the analog input with the voltage generated from D/A Converter, and output the comparison result to SAR and Control Logic Block for binary search. Three level amplifiers are employed in this comparator to provide enough gain.

## 46.4 Register Description

### 46.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
TSADC_USER_CON	0x0000	W	0x000000208	The control register of A/D Converter.
TSADC_AUTO_CON	0x0004	W	0x000000000	TSADC auto mode control register
TSADC_INT_EN	0x0008	W	0x000000000	Interrupt enable
TSADC_INT_PD	0x000c	W	0x000000000	Interrupt Status
TSADC_DATA0	0x0020	W	0x000000000	This register contains the data after A/D Conversion.
TSADC_DATA1	0x0024	W	0x000000000	This register contains the data after A/D Conversion.
TSADC_DATA2	0x0028	W	0x000000000	This register contains the data after A/D Conversion.
TSADC_DATA3	0x002c	W	0x000000000	This register contains the data after A/D Conversion.
TSADC_COMP0_INT	0x0030	W	0x000000000	TSADC high temperature level for source 0
TSADC_COMP1_INT	0x0034	W	0x000000000	TSADC high temperature level for source 1
TSADC_COMP2_INT	0x0038	W	0x000000000	TSADC high temperature level for source 2
TSADC_COMP3_INT	0x003c	W	0x000000000	TSADC high temperature level for source 3
TSADC_COMP0_SHU_T	0x0040	W	0x000000000	TSADC high temperature level for source 0
TSADC_COMP1_SHU_T	0x0044	W	0x000000000	TSADC high temperature level for source 1
TSADC_COMP2_SHU_T	0x0048	W	0x000000000	TSADC high temperature level for source 2
TSADC_COMP3_SHU_T	0x004c	W	0x000000000	TSADC high temperature level for source 3
TSADC_HIGHT_INT_DEBOUNCE	0x0060	W	0x000000003	high temperature debounce

Name	Offset	Size	Reset Value	Description
TSADC_HIGHT_TSH UT_DEBOUNCE	0x0064	W	0x00000003	high temperature debounce
TSADC_AUTO_PERIOD	0x0068	W	0x00010000	TSADC auto access period
TSADC_AUTO_PERIOD_HT	0x006c	W	0x00010000	TSADC auto access period when temperature is high

Notes: Size: **B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**-WORD (32 bits) access

#### 46.4.2 Detail Register Description

##### TSADC\_USER\_CON

Address: Operational Base + offset (0x0000)

The control register of A/D Converter.

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12	RO	0x0	adc_status ADC status (EOC) 0: ADC stop; 1: Conversion in progress.
11:6	RW	0x08	inter_pd_soc interleave between power down and start of conversion
5	RW	0x0	start When software write 1 to this bit , start_of_conversion will be assert. This bit will be cleared after TSADC access finishing.
4	RW	0x0	start_mode start mode. 0: tsadc controller will assert start_of_conversion after "inter_pd_soc" cycles. 1: the start_of_conversion will be controlled by TSADC_USER_CON[5].
3	RW	0x1	adc_power_ctrl ADC power down control bit 0: ADC power down; 1: ADC power up and reset.
2:0	RW	0x0	adc_input_src_sel ADC input source selection(CH_SEL[2:0]). 111 : Input source 0 (SARADC_AIN[0]) 110 : Input source 1 (SARADC_AIN[1]) 101 : Input source 2 (SARADC_AIN[2]) 100 : Input source 3 (SARADC_AIN[3]) Others : Reserved

**TSADC\_AUTO\_CON**

Address: Operational Base + offset (0x0004)

TSADC auto mode control register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:25	RO	0x0	reserved
24	RW	0x0	last_tshut TSHUT status. This bit will set to 1 when tshut is valid, and only be cleared when application write 1 to it. This bit will not be cleared by system reset.
23:18	RO	0x0	reserved
17	RO	0x0	sample_dly_sel 0: AUTO_PERIOD is used. 1: AUTO_PERIOD_HT is used.
16	RO	0x0	auto_status 0: auto mode stop; 1: auto mode in progress.
15:9	RO	0x0	reserved
8	RW	0x0	tshut polarity 0: low active 1: high active
7	RW	0x0	src3_en 0: do not care the temperature of source 3 1: if the temperature of source 3 is too high , TSHUT will be valid
6	RW	0x0	src2_en 0: do not care the temperature of source 2 1: if the temperature of source 2 is too high , TSHUT will be valid
5	RW	0x0	src1_en 0: do not care the temperature of source 1 1: if the temperature of source 1 is too high , TSHUT will be valid
4	RW	0x0	src0_en 0: do not care the temperature of source 0 1: if the temperature of source 0 is too high , TSHUT will be valid
3:1	RO	0x0	reserved
0	RW	0x0	auto_en 0: TSADC controller works at user-define mode 1: TSADC controller works at auto mode

**TSADC\_INT\_EN**

Address: Operational Base + offset (0x0008)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
------------	-------------	--------------------	--------------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:13	RO	0x0	reserved
12	RW	0x0	eoc_int_en eoc_Interrupt enable. eoc_interrupt enable in user defined mode 0: Disable; 1: Enable
11	RW	0x0	tshut_2cru_en_src3 0: TSHUT output to cru disabled. TSHUT output will always keep low . 1: TSHUT output works.
10	RW	0x0	tshut_2cru_en_src2 0: TSHUT output to cru disabled. TSHUT output will always keep low . 1: TSHUT output works.
9	RW	0x0	tshut_2cru_en_src1 0: TSHUT output to cru disabled. TSHUT output will always keep low . 1: TSHUT output works.
8	RW	0x0	tshut_2cru_en_src0 0: TSHUT output to cru disabled. TSHUT output will always keep low . 1: TSHUT output works.
7	RW	0x0	tshut_2gpio_en_src3 0: TSHUT output to gpio0b2 disabled. TSHUT output will always keep low . 1: TSHUT output works.
6	RW	0x0	tshut_2gpio_en_src2 0: TSHUT output to gpio0b2 disabled. TSHUT output will always keep low . 1: TSHUT output works.
5	RW	0x0	tshut_2gpio_en_src1 0: TSHUT output to gpio0b2 disabled. TSHUT output will always keep low . 1: TSHUT output works.
4	RW	0x0	tshut_2gpio_en_src0 0: TSHUT output to gpio0b2 disabled. TSHUT output will always keep low . 1: TSHUT output works.
3	RW	0x0	ht_inten_src3 high temperature interrupt enable for src3 0: disable 1: enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
2	RW	0x0	ht_inten_src2 high temperature interrupt enable for src2 0: disable 1: enable
1	RW	0x0	ht_inten_src1 high temperature interrupt enable for src1 0: disable 1: enable
0	RW	0x0	ht_inten_src0 high temperature interrupt enable for src0 0: disable 1: enable

**TSADC\_INT\_PD**

Address: Operational Base + offset (0x000c)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:9	RO	0x0	reserved
8	RW	0x0	eoc_int_pd Interrupt status. This bit will be set to 1 when end-of-conversion. Set 0 to clear the interrupt.
7	RW	0x0	tshut_o_src3 TSHUT output status
6	RW	0x0	tshut_o_src2 TSHUT output status
5	RW	0x0	tshut_o_src1 TSHUT output status
4	RW	0x0	tshut_o_src0 TSHUT output status
3	RW	0x0	ht_irq_src3 When TSADC output is smaller than COMP_INT, this bit will be valid, which means temperature is high, and the application should in charge of this. write 1 to it , this bit will be cleared.
2	RW	0x0	ht_irq_src2 When TSADC output is smaller than COMP_INT, this bit will be valid, which means temperature is high, and the application should in charge of this. write 1 to it , this bit will be cleared.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	ht_irq_src1 When TSADC output is smaller than COMP_INT, this bit will be valid, which means temperature is high, and the application should in charge of this. write 1 to it , this bit will be cleared.
0	RW	0x0	ht_irq_src0 When TSADC output is smaller than COMP_INT, this bit will be valid, which means temperature is high, and the application should in charge of this. write 1 to it , this bit will be cleared.

**TSADC\_DATA0**

Address: Operational Base + offset (0x0020)

This register contains the data after A/D Conversion.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RO	0x000	adc_data A/D value of the channel 0 last conversion (DOUT[9:0]).

**TSADC\_DATA1**

Address: Operational Base + offset (0x0024)

This register contains the data after A/D Conversion.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RO	0x000	adc_data A/D value of the channel 1 last conversion (DOUT[9:0]).

**TSADC\_DATA2**

Address: Operational Base + offset (0x0028)

This register contains the data after A/D Conversion.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RO	0x000	adc_data A/D value of the channel 2 last conversion (DOUT[9:0]).

**TSADC\_DATA3**

Address: Operational Base + offset (0x002c)

This register contains the data after A/D Conversion.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RO	0x000	adc_data A/D value of the channel 3 last conversion (DOUT[9:0]).

**TSADC\_COMPO\_INT**

Address: Operational Base + offset (0x0030)

TSADC high temperature level for source 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	tsadc_comp_src0 TSADC high temperature level. TSADC output is smaller than tsadc_comp, means the temperature is high. TSADC_INT will be valid.

**TSADC\_COMP1\_INT**

Address: Operational Base + offset (0x0034)

TSADC high temperature level for source 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	tsadc_comp_src1 TSADC high temperature level. TSADC output is smaller than tsadc_comp, means the temperature is high. TSADC_INT will be valid.

**TSADC\_COMP2\_INT**

Address: Operational Base + offset (0x0038)

TSADC high temperature level for source 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	tsadc_comp_src2 TSADC high temperature level. TSADC output is smaller than tsadc_comp, means the temperature is high. TSADC_INT will be valid.

**TSADC\_COMP3\_INT**

Address: Operational Base + offset (0x003c)

TSADC high temperature level for source 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	tsadc_comp_src3 TSADC high temperature level. TSADC output is smaller than tsadc_comp, means the temperature is high. TSADC_INT will be valid.

**TSADC\_COMPO\_SHUT**

Address: Operational Base + offset (0x0040)

TSADC high temperature level for source 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
11:0	RW	0x000	tsadc_comp_src0 TSADC high temperature level. TSADC output is smaller than tsadc_comp, means the temperature is too high. TSHUT will be valid.

**TSADC\_COMP1\_SHUT**

Address: Operational Base + offset (0x0044)

TSADC high temperature level for source 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	tsadc_comp_src1 TSADC high temperature level. TSADC output is smaller than tsadc_comp, means the temperature is too high. TSHUT will be valid.

**TSADC\_COMP2\_SHUT**

Address: Operational Base + offset (0x0048)

TSADC high temperature level for source 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	tsadc_comp_src2 TSADC high temperature level. TSADC output is smaller than tsadc_comp, means the temperature is too high. TSHUT will be valid.

**TSADC\_COMP3\_SHUT**

Address: Operational Base + offset (0x004c)

TSADC high temperature level for source 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11:0	RW	0x000	tsadc_comp_src3 TSADC high temperature level. TSADC output is smaller than tsadc_comp, means the temperature is too high. TSHUT will be valid.

**TSADC\_HIGHT\_INT\_DEBOUNCE**

Address: Operational Base + offset (0x0060)

high temperature debounce

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x03	debounce TSADC controller will only generate interrupt or TSHUT when temperature is higher than COMP_INT for "debounce" times.

**TSADC\_HIGHT\_TSHUT\_DEBOUNCE**

Address: Operational Base + offset (0x0064)

high temperature debounce

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x03	debounce TSADC controller will only generate interrupt or TSHUT when temperature is higher than COMP_SHUT for "debounce" times.

**TSADC\_AUTO\_PERIOD**

Address: Operational Base + offset (0x0068)

TSADC auto access period

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00010000	auto_period when auto mode is enabled, this register controls the interleave between every two accessing of TSADC.

**TSADC\_AUTO\_PERIOD\_HT**

Address: Operational Base + offset (0x006c)

TSADC auto access period when temperature is high

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00010000	auto_period This register controls the interleave between every two accessing of TSADC after the temperature is higher than COMP_SHUT or COMP_INT

## 46.5 Application Notes

### 46.5.1 Channel Select

The system has three Temperature Sensors, channel0 is reserve, and channel 1 is for CPU, and channel 2 is for GPU.

### 46.5.2 Single-sample conversion

To saving power, the TS-ADC used single-sample conversion. The timing as flowing picture:

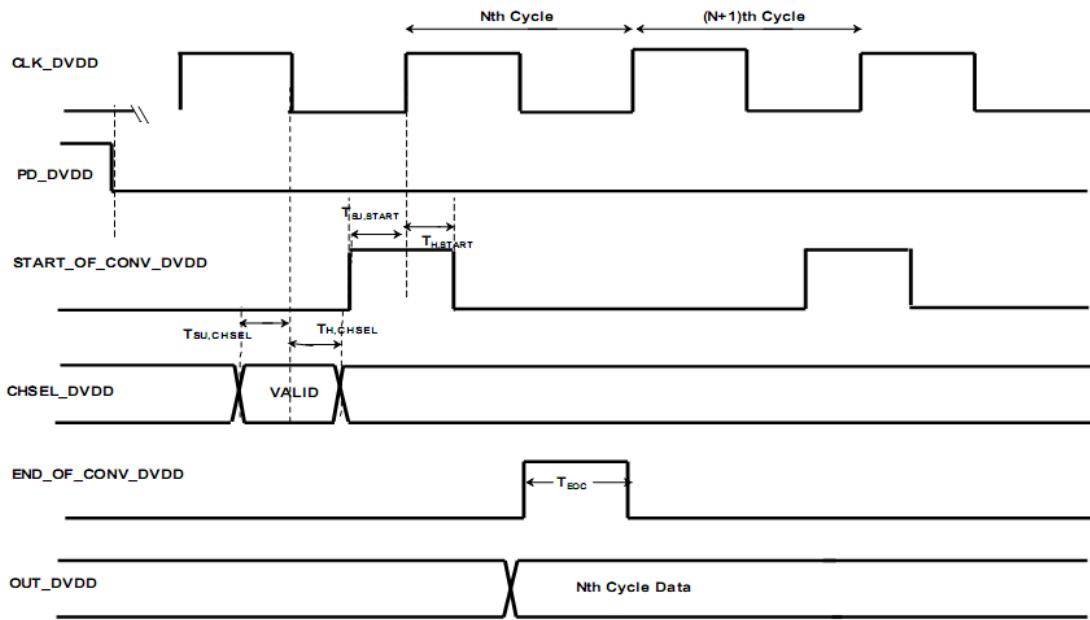


Fig. 46-2 Single-sample conversion



Fig. 46-3 Clock Timing Diagram

The below timing constraints are applicable for single-sample conversion mode.

Table 46-1 Timing parameters

Timing	Symbol	Value			Unit	Description
		Min	Typ	Max		
START_OF_CONV_Setup time	TSU,START	5			ns	Set Up time for START_OF_CONV w.r.t CLKIN rising edge
START_OF_CONV Hold time	TH,START	5			ns	Hold time for START_OF_CONV w.r.t CLKIN rising edge
CHSEL setup time	TSU,CHSEL	5			ns	Set Up time for CHSEL w.r.t CLKIN falling edge
CHSEL Hold time	TH,CHSEL	5			ns	Hold time for CHSEL w.r.t CLKIN falling edge
Data Setup	TSU,DATA	11		15	us	Set Up time for output data w.r.t either CLKIN rising edge or END_OF_CONV falling edge
Data Hold	TH,DATA	5		9	us	Hold time for output data w.r.t either CLKIN rising edge or END_OF_CONV falling edge
Data access time	TDAC	5		9	us	Valid data w.r.t CLKIN rising edge
Delay time	TDelay			5	ns	Delay between Valid data and EOC_DVDD rising edge
EOC Pulse Width (max frequency)	TEOC	11		15	us	Pulse width of EOC
CLKIN Rise Time	TCR			2	ns	CLKIN Rise Time
CLKIN Fall Time	TCF			2	ns	CLKIN Fall Time
CLK Pulse Width(Duty Cycle)	TCPW	45		55	%	CLKIN High/Low Time Period
CLK Period	TCP	20			us	CLKIN Time Period

Note: The time from negedge of PD\_DVDD to posedge of START\_OF\_CONV\_DVDD is more than 8 cycles of CLK\_DVDD, and less than 500us at same time.

### 46.5.3 Temperature-to-code mapping

Table 46-2 Temperature Code Mapping

temp (C)	Code
-40	3800
-35	3792
-30	3783
-25	3774
-20	3765
-15	3756
-10	3747
-5	3737
0	3728
5	3718
10	3708
15	3698
20	3688
25	3678
30	3667
35	3656
40	3645
45	3634
50	3623
55	3611
60	3600
65	3588
70	3575
75	3563
80	3550
85	3537
90	3524
95	3510
100	3496
105	3482
110	3467
115	3452
120	3437
125	3421

Note:

1. Code to Temperature mapping of the Temperature sensor is a piece wise linear curve. Any temperature, code falling between to 2 give temperatures can be linearly interpolated.
2. Code to Temperature mapping should be updated based on silicon results.

#### 46.5.4 User-Define Mode

1. In user-define mode, the PD\_DVDD and CHSEL\_DVDD are generate by setting register TSADC\_USER\_CON, bit[3] and bit[2:0]. In order to ensure timing between PD\_DVDD and CHSEL\_DVDD, the CHSEL\_DVDD must be set before the PD\_DVDD.
2. In user-define mode, you can choose the method to control the START\_OF\_CONVERSION by setting bit[4] of TSADC\_USER\_CON. If set to 0, the start\_of\_conversion will be assert after “inter\_pd\_soc” cycles, which could be set by bit[11:6] of TSADC\_USER\_CON. And if start\_mode was set 1, the start\_of\_conversion will be controlled by bit[5] of TSADC\_USER\_CON.
3. Software can get the four channel temperature from TSADC\_DATAAn (n=0,1,2,3).

#### 46.5.5 Automatic Mode

You can use the automatic mode with the following step:

1. Set TSADC\_AUTO\_PERIOD, configure the interleave between every two accessing of TSADC in normal operation.
2. Set TSADC\_AUTO\_PERIOD\_HT. configure the interleave between every two accessing of TSADC after the temperature is higher than COMP\_SHUT or COMP\_INT.
3. Set TSADC\_COMPn\_INT(n=0,1,2,3), configure the high temperature level, if tsadc output is smaller than the value, means the temperature is high, tsadc\_int will be asserted.
4. Set TSADC\_COMPn\_SHUT(n=0,1,2,3), configure the super high temperature level, if tsadc output is smaller than the value, means the temperature is too high, TSHUT will be asserted.
5. Set TSADC\_INT\_EN, you can enable the high temperature interrupt for all channel; and you can also set TSHUT output to gpio to reset the whole chip; and you can set TSHUT output to cru to reset the whole chip.
6. Set TSADC\_HIGHT\_INT\_DEBOUNCE and TSADC\_HIGHT\_TSHUT\_DEBOUNCE, if the temperature is higher than COMP\_INT or COMP\_SHUT for “debounce” times, TSADC controller will generate interrupt or TSHUT.
7. Set TSADC\_AUTO\_CON, enable the TSADC controller.

## Chapter 47 Timer

### 47.1 Overview

Timer is a programmable timer peripheral. This component is an APB slave device.

Timer0~4 and Timer6 count down from a programmed value and generate an interrupt when the count reaches zero.

Timer5 and Timer7 count up from zero to a programmed value and generate an interrupt when the count reaches the programmed value.

Timer supports the following features:

- Two APB timers in the soc system. One is in the alive subsystem, include two programmable 64 bits timer channel, acts as TIMER6 and TIMER7; The other is in the cpu subsystem, include six programmable 64 bits timer channel, acts as TIMER0, TIMER1, TIMER2, TIMER3, TIMER4, and TIMER5 respectively.
- Two operation modes: free-running and user-defined count.
- Maskable for each individual interrupt.
- TIMER5 is used for gpu ; TIMER7 is used for core.

### 47.2 Block Diagram

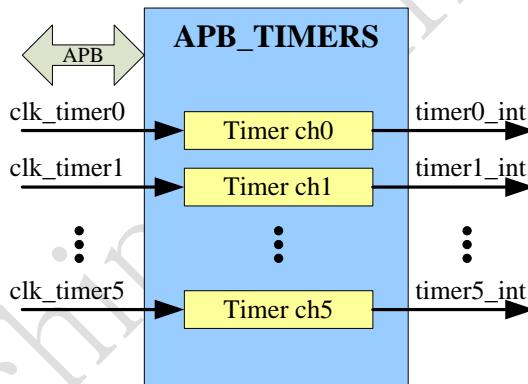


Fig. 47-1 Timers Block Diagram

The above figure shows the architecture of the APB timers (include six programmable timer channel) that in the cpu subsystem. The other APB timers that in the alive subsystem only include one programmable timer channel.

### 47.3 Function description

#### 47.3.1 Timer clock

TIMER0, TIMER1, TIMER2, Timer3, TIMER4 and TIMER5 are in the CPU subsystem, using timer ch0 ~ ch5 respectively. The timer clock is 24MHz OSC.

TIMER6 and TIMER7 are in the ALIVE subsystem, using timer ch0 ~ ch1. The timer clock is 24MHz OSC.

#### 47.3.2 Programming sequence

1. Initialize the timer by the TIMERn\_CONTROLREG ( $0 \leq n \leq 5$ ) register:

- Disable the timer by writing a "0" to the timer enable bit (bit 0). Accordingly, the timer\_en

- output signal is de-asserted.
  - Program the timer mode—user-defined or free-running—by writing a “0” or “1” respectively, to the timer mode bit (bit 1).
  - Set the interrupt mask as either masked or not masked by writing a “0” or “1” respectively, to the timer interrupt mask bit (bit 2).
2. Load the timer count value into the TIMERn\_LOAD\_COUNT1 ( $0 \leq n \leq 5$ ) and TIMERn\_LOAD\_COUNT0 ( $0 \leq n \leq 5$ ) register.
3. Enable the timer by writing a “1” to bit 0 of TIMERn\_CONTROLREG ( $0 \leq n \leq 5$ ).

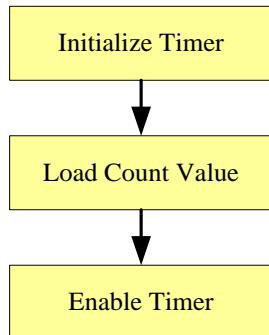


Fig. 47-2 Timer Usage Flow

### 47.3.3 Loading a timer count value

The initial value for each timer — that is, the value from which it counts down — is loaded into the timer using the load count register (TIMERn\_LOAD\_COUNT1 ( $0 \leq n \leq 5$ ) and TIMERn\_LOAD\_COUNT0 ( $0 \leq n \leq 5$ )). Two events can cause a timer to load the initial value from its load count register:

- Timer is enabled after reset or disabled.
- Timer counts down to 0, when timer is configured into free-running mode.

### 47.3.4 Timer mode selection

- User-defined count mode – Timer loads TIMERn\_LOAD\_COUNT1 ( $0 \leq n \leq 5$ ) and TIMERn\_LOAD\_COUNT0 ( $0 \leq n \leq 5$ ) register as initial value. Timer will not automatically load the count register, when timer counts down to 0. User need to disable timer firstly and follow the programming sequence to make timer work again.
- Free-running mode – Timer loads the TIMERn\_LOAD\_COUNT1 ( $0 \leq n \leq 5$ ) and TIMERn\_LOAD\_COUNT0 ( $0 \leq n \leq 5$ ) register as initial value. Timer will automatically load the count register, when timer counts down to 0.

## 47.4 Register Description

This section describes the control/status registers of the design. Software should read and write these registers using 32-bits accesses.

### 47.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
TIMER0_LOAD_COUNT0	0x0000	W	0x00000000	Timer0 Load Count Register
TIMER0_LOAD_COUNT1	0x0004	W	0x00000000	Timer0 Load Count Register
TIMER0_CURRENT_VALUE0	0x0008	W	0x00000000	Timer0 Current Value Register
TIMER0_CURRENT_VALUE1	0x000C	W	0x00000000	Timer0 Current Value Register
TIMER0_CONTROLREG	0x0010	W	0x00000000	Timer0 Control Register

TIMER0_INTSTATUS	0x0018	W	0x00000000	Timer0 Interrupt Status Register
TIMER1_LOAD_COUNT0	0x0020	W	0x00000000	Timer1 Load Count Register
TIMER1_LOAD_COUNT1	0x0024	W	0x00000000	Timer1 Load Count Register
TIMER1_CURRENT_VALUE0	0x0028	W	0x00000000	Timer1 Current Value Register
TIMER1_CURRENT_VALUE1	0x002c	W	0x00000000	Timer1 Current Value Register
TIMER1_CONTROLREG	0x0030	W	0x00000000	Timer1 Control Register
TIMER1_INTSTATUS	0x0038	W	0x00000000	Timer1 Interrupt Status Register
TIMER2_LOAD_COUNT0	0x0040	W	0x00000000	Timer2 Load Count Register
TIMER2_LOAD_COUNT1	0x0044	W	0x00000000	Timer2 Load Count Register
TIMER2_CURRENT_VALUE0	0x0048	W	0x00000000	Timer2 Current Value Register
TIMER2_CURRENT_VALUE1	0x004c	W	0x00000000	Timer2 Current Value Register
TIMER2_CONTROLREG	0x0050	W	0x00000000	Timer2 Control Register
TIMER2_INTSTATUS	0x0058	W	0x00000000	Timer2 Interrupt Status Register
TIMER3_LOAD_COUNT0	0x0060	W	0x00000000	Timer3 Load Count Register
TIMER3_LOAD_COUNT1	0x0064	W	0x00000000	Timer3 Load Count Register
TIMER3_CURRENT_VALUE0	0x0068	W	0x00000000	Timer3 Current Value Register
TIMER3_CURRENT_VALUE1	0x006c	W	0x00000000	Timer3 Current Value Register
TIMER3_CONTROLREG	0x0070	W	0x00000000	Timer3 Control Register
TIMER3_INTSTATUS	0x0078	W	0x00000000	Timer3 Interrupt Status Register
TIMER4_LOAD_COUNT0	0x0080	W	0x00000000	Timer4 Load Count Register
TIMER4_LOAD_COUNT1	0x0084	W	0x00000000	Timer4 Load Count Register
TIMER4_CURRENT_VALUE0	0x0088	W	0x00000000	Timer4 Current Value Register
TIMER4_CURRENT_VALUE1	0x008c	W	0x00000000	Timer4 Current Value Register
TIMER4_CONTROLREG	0x0090	W	0x00000000	Timer4 Control Register
TIMER4_INTSTATUS	0x0098	W	0x00000000	Timer4 Interrupt Status Register
TIMER5_LOAD_COUNT0	0x00a0	W	0x00000000	Timer5 Load Count Register
TIMER5_LOAD_COUNT1	0x00a4	W	0x00000000	Timer5 Load Count Register
TIMER5_CURRENT_VALUE0	0x00a8	W	0x00000000	Timer5 Current Value Register
TIMER5_CURRENT_VALUE1	0x00ac	W	0x00000000	Timer5 Current Value Register
TIMER5_CONTROLREG	0x00b0	W	0x00000000	Timer5 Control Register
TIMER5_INTSTATUS	0x00b8	W	0x00000000	Timer5 Interrupt Status Register

Notes: Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

#### 47.4.2 Detail Register Description

##### **TIMERn\_LOAD\_COUNT0**

Address: Operational Base + offset(0x00+n\*0x20)

Timer n Load Count Register ( $0 \leq n \leq 5$ )

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Low 32 bits Value to be loaded into Timer n. This is the value from which counting commences.

##### **TIMERn\_LOAD\_COUNT1**

Address: Operational Base + offset(0x04+n\*0x20)

Timer n Load Count Register ( $0 \leq n \leq 5$ )

Bit	Attr	Reset Value	Description
31:0	RW	0x0	High 32 bits Value to be loaded into Timer n. This is the value from which counting commences.

**TIMERn\_CURRENT\_VALUE0**

Address: Operational Base + offset(0x08+n\*0x20)

Timer n Current Value Register ( $0 \leq n \leq 5$ )

Bit	Attr	Reset Value	Description
31:0	R	0x0	Low 32 bits of Current Value of Timer n.

**TIMERn\_CURRENT\_VALUE1**

Address: Operational Base + offset(0x0c+n\*0x20)

Timer n Current Value Register ( $0 \leq n \leq 5$ )

Bit	Attr	Reset Value	Description
31:0	R	0x0	High 32 bits of Current Value of Timer n.

**TIMERn\_CONTROLREG**

Address: Operational Base + offset(0x10+n\*0x20)

Timer n Control Register ( $0 \leq n \leq 5$ )

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Timer interrupt mask. 1'b0: mask 1'b1: not mask
1	RW	0x0	Timer mode. 1'b0: free-running mode 1'b1: user-defined count mode
0	RW	0x0	Timer enable. 1'b0: disable 1'b1: enable

**TIMERn\_INTSTATUS**

Address: Operational Base + offset(0x18+n\*0x20)

Timer n Interrupt Status Register ( $0 \leq n \leq 5$ )

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	This register contains the interrupt status for timer n Write 1 to this register will clear the interrupt

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

## 47.5 Application Notes

In the chip, the timer\_clk is from 24MHz OSC, asynchronous to the pclk. When user disable the timer enable bit (bit 0 of TIMERn\_CONTROLREG ( $0 \leq n \leq 5$ )), the timer\_en output signal is de-asserted, and timer\_clk will stop. When user enable the timer, the timer\_en signal is asserted and timer\_clk will start running.

The application is only allowed to re-config registers when timer\_en is low.

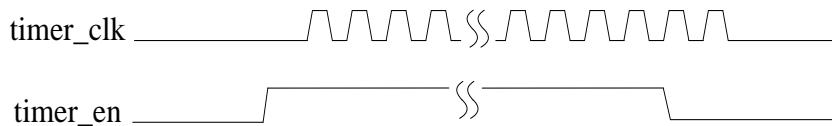


Fig. 47-3 Timing between timer\_en and timer\_clk

Please refer to funciton description section for the timer usage flow.

## Chapter 48 eFuse

### 48.1 Overview

In RK3288, there are two eFuse. One is organized as 32bits by 8 one-time programmable electrical fuses with random access interface, and the other is organized as 32bits by 32 one-time programmable electrical fuses.

The 32x32 eFuse can only be accessed by APB bus when IO\_SECURITYsel is high.

It is a type of non-volatile memory fabricated in standard CMOS logic process. The main features are as follows:

- Programming condition :  $V_{QPS\_EFUSE} = 1.5V \pm 10\%$
- Program time :  $10\mu s \pm 0.2\mu s$  .
- Read condition :  $V_{QPS\_EFUSE} = 0V$
- Provide standby mode

### 48.2 Block Diagram

In the following diagram, all the signals except power supply VDD\_EFUSE and VQPS\_EFUSE are controlled by registers. For detailed description, please refer to detailed register descriptions.

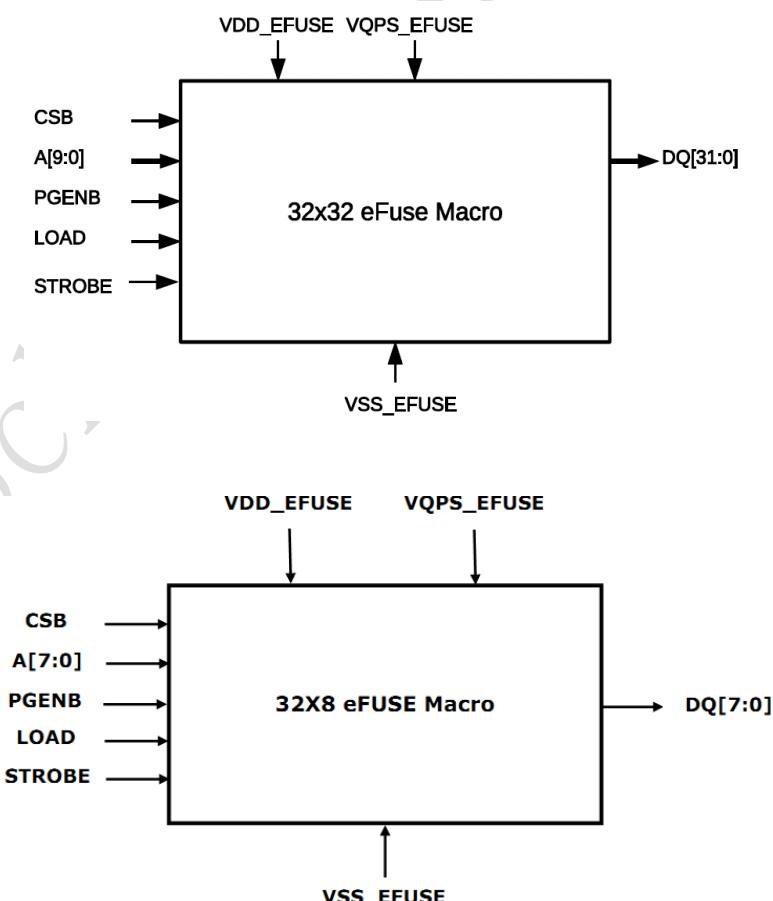


Fig. 48-1 RK3288 eFuse block diagram

## 48.3 Function description

eFuse has three operation modes. They are defined as standby, read and programming.

### Program (PGM) Mode

In order to enter programming mode, the following conditions need to be satisfied: VQPS\_EFUSE is at high voltage, LOAD signal is low, PGENB signal is low, and CSB signal is low. All bits can be individually programmed (one at a time) with the proper address selected, the STROBE signal high and the address bits satisfying setup and hold time with respect to STROBE.

### Read Mode

In order to enter read mode the following conditions need to be satisfied: VQPS\_EFUSE is at ground, the LOAD signal is high, the PGENB signal is high, and the CSB is low. An entire 8-bit word of data can be read in one read operation with STROBE being high and a proper address selected (address signals A5~A7 are "don't cares").

### Standby Mode

Standby is defined when the macro is not being programmed or read. The conditions for standby mode are: the LOAD signal is low, the STROBE signal is low, the CSB signal is high and PGENB is high.

## 48.4 Register Description

This section describes the control/status registers of the design.

### 48.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
EFUSE_CTRL	0x0000	W	0x00000000	efuse control register
EFUSE_DOUT	0x0004	W	0x00000000	efuse data out register

Notes: Size: **B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**- WORD (32 bits) access

### 48.4.2 Detail Register Description

#### EFUSE\_CTRL

Address: Operational Base + offset (0x0000)

eFuse control register

Bit	Attr	Reset Value	Description
31:14	RO	0x0	reserved
15:6	RW	0x00	efuse_addr efuse address pins : A[7:0] / A[9:0]
5:4	RO	0x0	reserved
3	RW	0x0	efuse_pgenb efuse program enable (active low) : PGENB
2	RW	0x0	efuse_load efuse turn on sense amplifier and load data into latch (active high) : LOAD

Bit	Attr	Reset Value	Description
1	RW	0x0	efuse_strobe efuse turn on the array for read or program access (active high) : STROBE
0	RW	0x0	efuse_csb efuse chip select enable signal, active low : CSB

**EFUSE\_DOUT**

Address: Operational Base + offset (0x0004)

eFuse data out register

Bit	Attr	Reset Value	Description
31:0	RO	0x00	efuse_dout efuse data output

**48.5 Timing Diagram**

- When efuse is in program(PGM) mode.

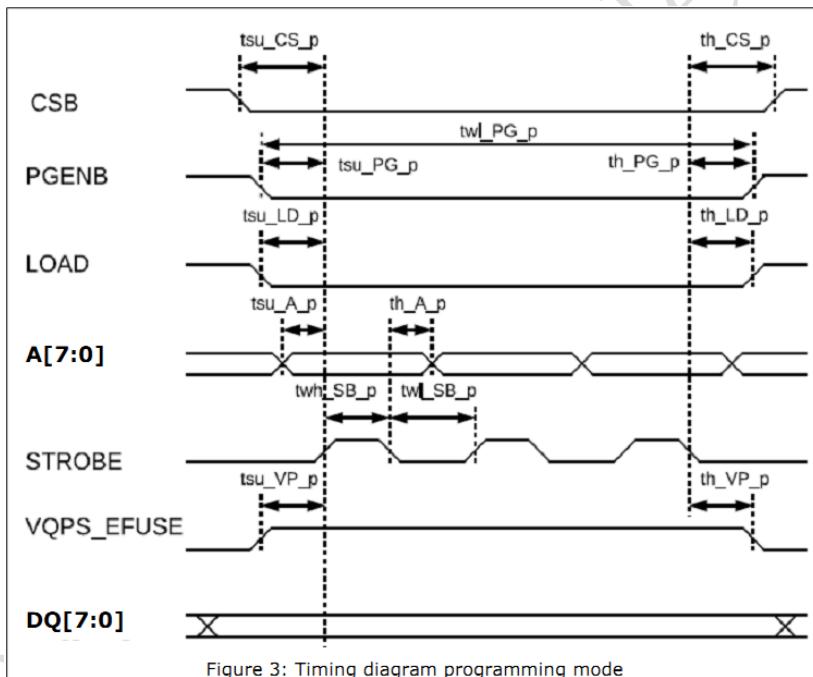


Fig. 48-2 RK3288 efuse timing diagram in program mode

- When efuse is in read mode.

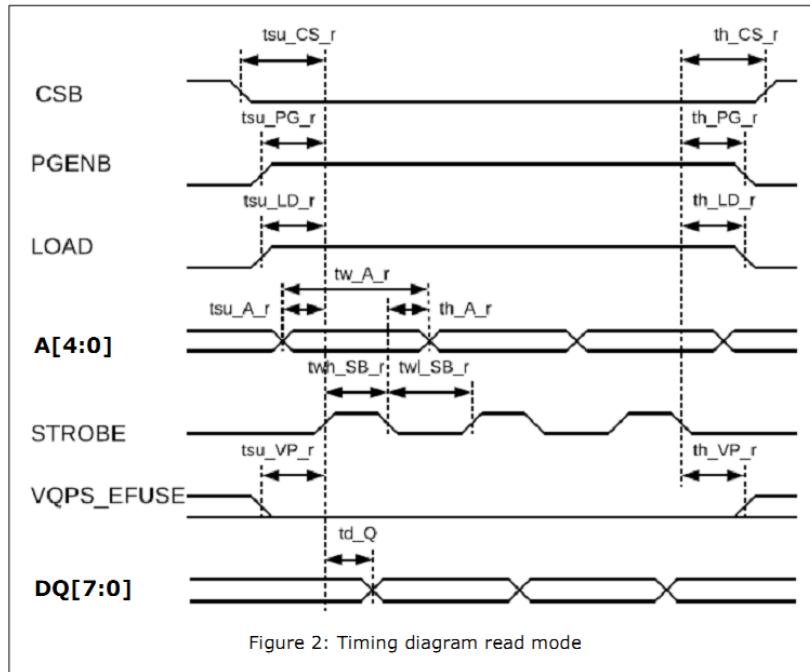


Fig. 48-3 RK3288 efuse timing diagram in read mode

The following table has shows the detailed value for timing parameters in the above diagram.

Table 48-1 RK3288 eFuse timing parameters list

Mode	Item	Description	Min	Typ	Max	Unit
Read Mode	twh_SB_r	Pulse width high of STROBE read strobe	20	-	-	ns
	twl_SB_r	Pulse width low of STROBE read strobe	15	-	-	ns
	tsu_A_r	A[7:0] to STROBE setup time in read mode	25	-	-	ns
	th_A_r	A[7:0] to STROBE hold time in read mode	3	-	-	ns
	tw_A_r	A[7:0] pulse width while LOAD high in read mode	48	-	100	ns
	tsu_CS_r	CSB to STROBE setup time in read mode	16	-	-	ns
	th_CS_r	CSB to STROBE hold time in read mode	6	-	-	ns
	tsu_PG_r	PGENB to STROBE setup time in read mode	14	-	-	ns
	th_PG_r	PGENB to STROBE hold time in read mode	10	-	-	ns
	tsu_LD_r	LOAD to STROBE setup time in read mode	10	-	-	ns
	th_LD_r	LOAD to STROBE hold time in read mode	7	-	-	ns
	tsu_VP_r	VQPS_EFUSE to STROBE setup time in read mode	20	-	-	ns
	th_VP_r	VQPS_EFUSE to STROBE hold time in read mode	20	-	-	ns
	td_Q	DQ[7:0] delay time after STROBE high	0	-	8	ns
PGM Mode	twh_SB_p	Pulse width high of STROBE PGM strobe	9.8	10	10.2	us
	twl_SB_p	Pulse width low of STROBE PGM strobe	15	-	-	ns
	tsu_A_p	A[7:0] to STROBE setup time in PGM mode	12	-	-	ns
	th_A_p	A[7:0] to STROBE hold time in PGM mode	3	-	-	ns
	tsu_CS_p	CSB to STROBE setup time in PGM mode	16	-	-	ns
	th_CS_p	CSB to STROBE hold time in PGM mode	6	-	-	ns
	tsu_PG_p	PGENB to STROBE setup time in PGM mode	14	-	-	ns
	th_PG_p	PGENB to STROBE hold time in PGM mode	10	-	-	ns
	twl_PG_p	PGENB pulse width low (cumulative) in PGM mode	-	-	100	ms
	tsu_LD_p	LOAD to STROBE setup time in PGM mode	10	-	-	ns
	th_LD_p	LOAD to STROBE hold time in PGM mode	7	-	-	ns
	tsu_VP_p	VQPS_EFUSE to STROBE setup time in PGM mode	20	-	-	ns
	th_VP_p	VQPS_EFUSE to STROBE hold time in PGM mode	20	-	-	ns

## **48.6 Application Notes**

During usage of efuse, customers must pay more attention to the following items:

1. In condition of program(PGM) mode, VQPS\_EFUSE=  $1.5V \pm 10\%$ .
2. Q0~Q7/Q31 will be reset to “0” once CSB at high.
3. No data access allowed at the rising edge of CSB.
4. All the program timing for each signal must be more than the value defined in the timing table.

## Chapter 49 General-Purpose Input/Output Ports (GPIO)

### 49.1 Overview

GPIO is a programmable General Purpose Programming I/O peripheral. This component is an APB slave device. GPIO controls the output data and direction of external I/O pads. It also can read back the data on external pads using memory-mapped registers.

GPIO supports the following features:

- 32 bits APB bus width
- 32 independently configurable signals
- Separate data registers and data direction registers for each signal
- Software control for each signal, or for each bit of each signal
- Configurable interrupt mode

### 49.2 Block Diagram

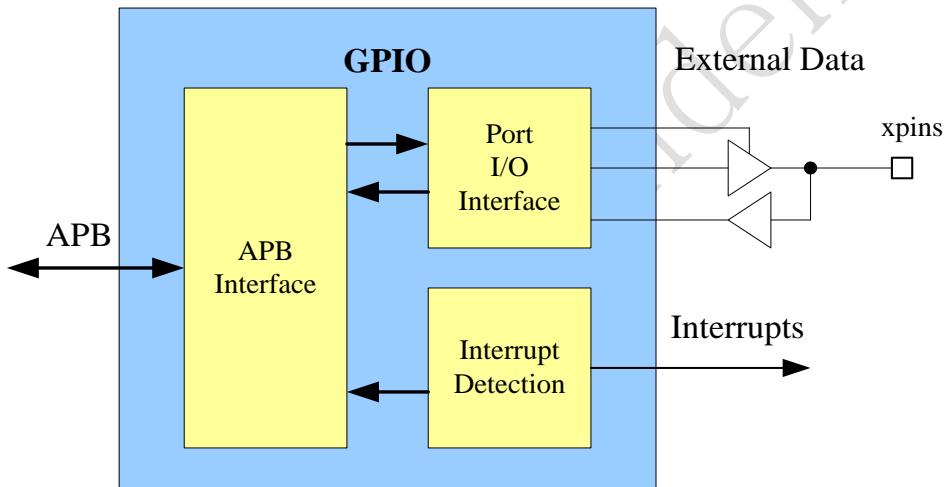


Fig. 49-1 GPIO block diagram

#### Block descriptions:

##### APB Interface

The APB Interface implements the APB slave operation. Its data bus width is 32 bits.

##### Port I/O Interface

External data Interface to or from I/O pads.

##### Interrupt Detection

Interrupt interface to or from interrupt controller.

### 49.3 Function description

#### 49.3.1 Operation

##### Control Mode (software)

Under software control, the data and direction control for the signal are sourced from the data register (GPIO\_SWPORTA\_DR) and direction control register (GPIO\_SWPORTA\_DDR).

The direction of the external I/O pad is controlled by a write to the Porta data direction register (GPIO\_SWPORTA\_DDR). The data written to this memory-mapped register gets mapped onto an output signal, GPIO\_PORTA\_DDR, of the GPIO peripheral. This output signal controls the direction of an external I/O pad.

The data written to the Porta data register (GPIO\_SWPORTA\_DR) drives the output buffer of the I/O pad. External data are input on the external data signal, GPIO\_EXT\_PORTA. Reading the external signal register (GPIO\_EXT\_PORTA) shows the value on the signal, regardless of the direction. This register is read-only, meaning that it cannot be written from the APB software interface.

### **Reading External Signals**

The data on the GPIO\_EXT\_PORTA external signal can always be read. The data on the external GPIO signal is read by an APB read of the memory-mapped register, GPIO\_EXT\_PORTA.

An APB read to the GPIO\_EXT\_PORTA register yields a value equal to that which is on the GPIO\_EXT\_PORTA signal.

### **Interrupts**

Port A can be programmed to accept external signals as interrupt sources on any of the bits of the signal. The type of interrupt is programmable with one of the following settings:

- Active-high and level
- Active-low and level
- Rising edge
- Falling edge

The interrupts can be masked by programming the GPIO\_INTMASK register. The interrupt status can be read before masking (called raw status) and after masking.

The interrupts are combined into a single interrupt output signal, which has the same polarity as the individual interrupts. In order to mask the combined interrupt, all individual interrupts have to be masked. The single combined interrupt does not have its own mask bit.

Whenever Port A is configured for interrupts, the data direction must be set to Input. If the data direction register is reprogrammed to Output, then any pending interrupts are not lost. However, no new interrupts are generated.

For edge-detected interrupts, the ISR can clear the interrupt by writing a 1 to the GPIO\_PORTA\_EOI register for the corresponding bit to disable the interrupt. This write also clears the interrupt status and raw status registers. Writing to the GPIO\_PORTA\_EOI register has no effect on level-sensitive interrupts. If level-sensitive interrupts cause the processor to interrupt, then the ISR can poll the GPIO\_INT\_RAWSTATUS register until the interrupt source disappears, or it can write to the GPIO\_INTMASK register to mask the interrupt before exiting the ISR. If the ISR exits without masking or disabling the interrupt prior to exiting, then the level-sensitive interrupt repeatedly requests an interrupt until the interrupt is cleared at the source.

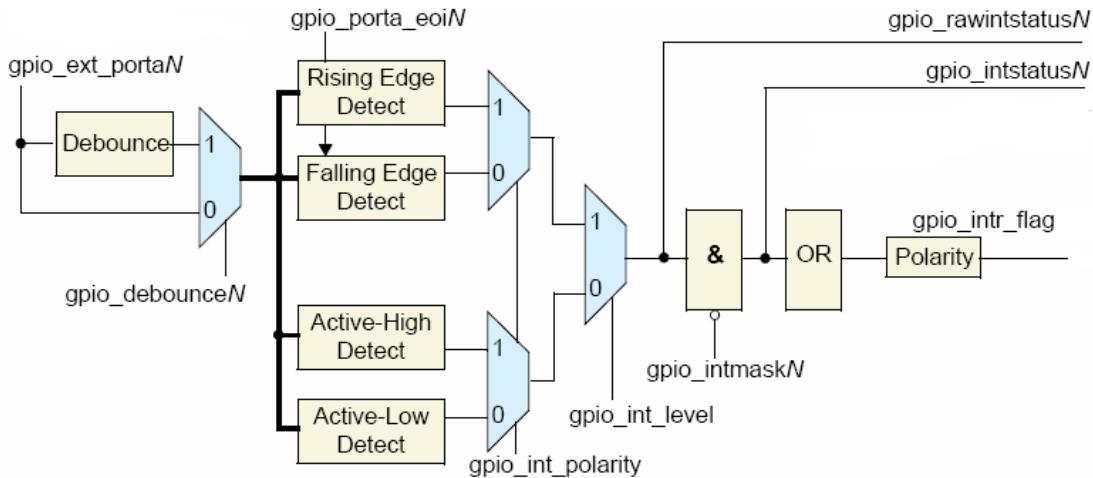


Fig. 49-2 GPIO Interrupt RTL Block Diagram

### Debounce operation

Port A has been configured to include the debounce capability interrupt feature. The external signal can be debounced to remove any spurious glitches that are less than one period of the external debouncing clock.

When input interrupt signals are debounced using a debounce clock (`pclk`), the signals must be active for a minimum of two cycles of the debounce clock to guarantee that they are registered. Any input pulse widths less than a debounce clock period are bounced. A pulse width between one and two debounce clock widths may or may not propagate, depending on its phase relationship to the debounce clock. If the input pulse spans two rising edges of the debounce clock, it is registered. If it spans only one rising edge, it is not registered.

### Synchronization of Interrupt Signals to the System Clock

Interrupt signals are internally synchronized to `pclk`. Synchronization to `pclk` must occur for edge-detect signals. With level-sensitive interrupts, synchronization is optional and under software control (`GPIO_LS_SYNC`).

### 49.3.2 Programming

#### Programming Considerations

- Reading from an unused location or unused bits in a particular register always returns zeros. There is no error mechanism in the APB.
- Programming the GPIO registers for interrupt capability, edge-sensitive or level-sensitive interrupts, and interrupt polarity should be completed prior to enabling the interrupts on Port A in order to prevent spurious glitches on the interrupt lines to the interrupt controller.
- Writing to the interrupt clear register clears an edge-detected interrupt and has no effect on a level-sensitive interrupt.

#### 9 GPIOs' hierarchy in the chip

`GPIO0` is in `PD_PMU` subsystem, `GPIO1~8` are in `PD_ALIVE` subsystem.

### 49.4 Register Description

This section describes the control/status registers of the design. Software should read and write these registers using 32-bits accesses. There are 9 GPIOs (`GPIO0 ~ GPIO8`), and each of them has same register group. Therefore, 9 GPIOs' register groups have 9 different base

address.

#### 49.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
GPIO_SWPORTA_DR	0x0000	W	0x00000000	Port A data register
GPIO_SWPORTA_DD R	0x0004	W	0x00000000	Port A data direction register
GPIO_INTEN	0x0030	W	0x00000000	Interrupt enable register
GPIO_INTMASK	0x0034	W	0x00000000	Interrupt mask register
GPIO_INTPOLTYPE_LEVEL	0x0038	W	0x00000000	Interrupt level register
GPIO_INT_POLARITY	0x003c	W	0x00000000	Interrupt polarity register
GPIO_INT_STATUS	0x0040	W	0x00000000	Interrupt status of port A
GPIO_INT_RAWSTATUS	0x0044	W	0x00000000	Raw Interrupt status of port A
GPIO_DEBOUNCE	0x0048	W	0x00000000	Debounce enable register
GPIO_PORTA_EOI	0x004c	W	0x00000000	Port A clear interrupt register
GPIO_EXT_PORTA	0x0050	W	0x00000000	Port A external port register
GPIO_LS_SYNC	0x0060	W	0x00000000	Level_sensitive synchronization enable register

Notes: *Size* : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

#### 49.4.2 Detail Register Description

##### GPIO\_SWPORTA\_DR

Address: Operational Base + offset (0x0000)

Port A data register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	gpio_swporta_dr Values written to this register are output on the I/O signals for Port A if the corresponding data direction bits for Port A are set to Output mode. The value read back is equal to the last value written to this register.

##### GPIO\_SWPORTA\_DDR

Address: Operational Base + offset (0x0004)

Port A data direction register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	<p>gpio_swporta_ddr</p> <p>Values written to this register independently control the direction of the corresponding data bit in Port A.</p> <p>1'b0: Input (default)</p> <p>1'b1: Output</p>

**GPIO\_INTEN**

Address: Operational Base + offset (0x0030)

Interrupt enable register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	<p>gpio_int_en</p> <p>Allows each bit of Port A to be configured for interrupts. Whenever a 1 is written to a bit of this register, it configures the corresponding bit on Port A to become an interrupt; otherwise, Port A operates as a normal GPIO signal.</p> <p>Interrupts are disabled on the corresponding bits of Port A if the corresponding data direction register is set to Output.</p> <p>1'b0: Configure Port A bit as normal GPIO signal (default)</p> <p>1'b1: Configure Port A bit as interrupt</p>

**GPIO\_INTMASK**

Address: Operational Base + offset (0x0034)

Interrupt mask register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	<p>gpio_int_mask</p> <p>Controls whether an interrupt on Port A can create an interrupt for the interrupt controller by not masking it. Whenever a 1 is written to a bit in this register, it masks the interrupt generation capability for this signal; otherwise interrupts are allowed through.</p> <p>1'b0: Interrupt bits are unmasked (default)</p> <p>1'b1: Mask interrupt</p>

**GPIO\_INTPTYPE\_LEVEL**

Address: Operational Base + offset (0x0038)

Interrupt level register

Bit	Attr	Reset Value	Description

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	gpio_inttype_level Controls the type of interrupt that can occur on Port A. 1'b0: Level-sensitive (default) 1'b1: Edge-sensitive

**GPIO\_INT\_POLARITY**

Address: Operational Base + offset (0x003c)

Interrupt polarity register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	gpio_int_polarity Controls the polarity of edge or level sensitivity that can occur on input of Port A. 1'b0: Active-low (default) 1'b1: Active-high

**GPIO\_INT\_STATUS**

Address: Operational Base + offset (0x0040)

Interrupt status of port A

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	gpio_int_status Interrupt status of Port A

**GPIO\_INT\_RAWSTATUS**

Address: Operational Base + offset (0x0044)

Raw Interrupt status of port A

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	gpio_int_rawstatus Raw interrupt of status of Port A (premasking bits)

**GPIO\_DEBOUNCE**

Address: Operational Base + offset (0x0048)

Debounce enable register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	gpio_debounce Controls whether an external signal that is the source of an interrupt needs to be debounced to remove any spurious glitches. Writing a 1 to a bit in this register enables the debouncing circuitry. A signal must be valid for two periods of an external clock before it is internally processed. 1'b0: No debounce (default) 1'b1: Enable debounce

**GPIO\_PORTA\_EOI**

Address: Operational Base + offset (0x004c)

Port A clear interrupt register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	WO	0x00000000	gpio_porta_eoi Controls the clearing of edge type interrupts from Port A. When a 1 is written into a corresponding bit of this register, the interrupt is cleared. All interrupts are cleared when Port A is not configured for interrupts. 1'b0: No interrupt clear (default) 1'b1: Clear interrupt

**GPIO\_EXT\_PORTA**

Address: Operational Base + offset (0x0050)

Port A external port register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	gpio_ext_porta When Port A is configured as Input, then reading this location reads the values on the signal. When the data direction of Port A is set as Output, reading this location reads the data register for Port A.

**GPIO\_LS\_SYNC**

Address: Operational Base + offset (0x0060)

Level\_sensitive synchronization enable register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	gpio_ls_sync Writing a 1 to this register results in all level-sensitive interrupts being synchronized to pclk_intr. 1'b0: No synchronization to pclk_intr (default) 1'b1: Synchronize to pclk_intr

**49.5 Interface description**

Table 49-1 GPIO interface description

<b>Module Pin</b>	<b>Dir</b>	<b>Pad Name</b>	<b>IOMUX Setting</b>
GPIO0 Interface			
gpio0_porta[7:0]	I/O	GPIO0_A[7:0]	GRF_GPIO0A_IOMUX[15:0] =16'h0000
gpio0_porta[15:8]	I/O	GPIO0_B[7:0]	GRF_GPIO0B_IOMUX[15:0] =16'h0000
gpio0_porta[23:16]	I/O	GPIO0_C[7:0]	GRF_GPIO0C_IOMUX[15:0]

			=16'h0000
gpio0_porta[31:24]	I/O	GPIO0_D[7:0]	GRF_GPIO0D_IOMUX[15:0] =16'h0000
GPIO1 Interface			
gpio1_porta[7:0]	I/O	GPIO1_A[7:0]	GRF_GPIO1A_IOMUX[15:0] =16'h0000
gpio1_porta[15:8]	I/O	GPIO1_B[7:0]	GRF_GPIO1B_IOMUX[15:0] =16'h0000
gpio1_porta[23:16]	I/O	GPIO1_C[7:0]	GRF_GPIO1C_IOMUX[15:0] =16'h0000
gpio1_porta[31:24]	I/O	GPIO1_D[7:0]	GRF_GPIO1D_IOMUX[15:0] =16'h0000
GPIO2 Interface			
gpio2_porta[7:0]	I/O	GPIO2_A[7:0]	GRF_GPIO2A_IOMUX[15:0] =16'h0000
gpio2_porta[15:8]	I/O	GPIO2_B[7:0]	GRF_GPIO2B_IOMUX[15:0] =16'h0000
gpio2_porta[23:16]	I/O	GPIO2_C[7:0]	GRF_GPIO2C_IOMUX[15:0] =16'h0000
gpio2_porta[31:24]	I/O	GPIO2_D[7:0]	GRF_GPIO2D_IOMUX[15:0] =16'h0000
GPIO3 Interface			
gpio3_porta[7:0]	I/O	GPIO3_A[7:0]	GRF_GPIO3A_IOMUX[15:0] =16'h0000
gpio3_porta[15:8]	I/O	GPIO3_B[7:0]	GRF_GPIO3B_IOMUX[15:0] =16'h0000
gpio3_porta[23:16]	I/O	GPIO3_C[7:0]	GRF_GPIO3C_IOMUX[15:0] =16'h0000
gpio3_porta[31:24]	I/O	GPIO3_D[7:0]	GRF_GPIO3D_IOMUX[15:0] =16'h0000

## 49.6 Application Notes

### Steps to set GPIO's direction

- Write GPIO\_SWPORT\_DDR[x] as 1 to set this gpio as output direction and Write GPIO\_SWPORT\_DDR[x] as 0 to set this gpio as input direction.
- Default GPIO's direction is input direction.

### Steps to set GPIO's level

- Write GPIO\_SWPORT\_DDR[x] as 1 to set this gpio as output direction.
- Write GPIO\_SWPORT\_DR[x] as v to set this GPIO's value.

### Steps to get GPIO's level

- Write GPIO\_SWPORT\_DDR[x] as 0 to set this gpio as input direction.
- Read from GPIO\_EXT\_PORT[x] to get GPIO's value

### Steps to set GPIO as interrupt source

- Write GPIO\_SWPORT\_DDR[x] as 0 to set this gpio as input direction.

- Write GPIO\_INTPOL[x] as v1 and write GPIO\_INT\_Polarity[x] as v2 to set interrupt type
- Write GPIO\_INTEN[x] as 1 to enable GPIO's interrupt

Note: Please switch iomux to GPIO mode first!

Rockchip Confidential

## Chapter 50 WatchDog

### 50.1 Overview

Watchdog Timer (WDT) is an APB slave peripheral that can be used to prevent system lockup that may be caused by conflicting parts or programs in a SoC. The WDT would generate interrupt or reset signal when its counter reaches zero, then a reset controller would reset the system.

WDT supports the following features:

- 32 bits APB bus width
- WDT counter's clock is pclk
- 32 bits WDT counter width
- Counter counts down from a preset value to 0 to indicate the occurrence of a timeout
- WDT can perform two types of operations when timeout occurs:
  - Generate a system reset
  - First generate an interrupt and if this is not cleared by the service routine by the time a second timeout occurs then generate a system reset
- Programmable reset pulse length
- Total 16 defined-ranges of main timeout period

### 50.2 Block Diagram

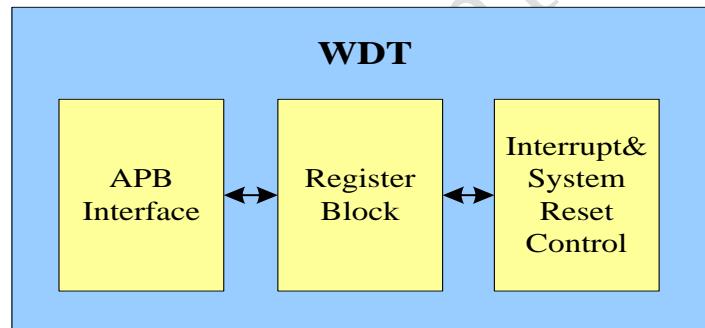


Fig. 50-1 WDT block diagram

#### Block Descriptions:

##### APB Interface

The APB Interface implements the APB slave operation. Its data bus width is 32 bits.

##### Register Block

A register block that reads coherence for the current count register.

##### Interrupt & system reset control

An interrupt/system reset generation block comprising of a decrementing counter and control logic.

### 50.3 Function description

#### 50.3.1 Operation

##### Counter

The WDT counts from a preset (timeout) value in descending order to zero. When the counter reaches zero, depending on the output response mode selected, either a system reset or an

interrupt occurs. When the counter reaches zero, it wraps to the selected timeout value and continues decrementing. The user can restart the counter to its initial value. This is programmed by writing to the restart register at any time. The process of restarting the watchdog counter is sometimes referred as kicking the dog. As a safety feature to prevent accidental restarts, the value 0x76 must be written to the Current Counter Value Register (WDT\_CRR).

## **Interrupts**

The WDT can be programmed to generate an interrupt (and then a system reset) when a timeout occurs. When a 1 is written to the response mode field (RMOD, bit 1) of the Watchdog Timer Control Register (WDT\_CR), the WDT generates an interrupt. If it is not cleared by the time a second timeout occurs, then it generates a system reset. If a restart occurs at the same time the watchdog counter reaches zero, an interrupt is not generated.

## **System Resets**

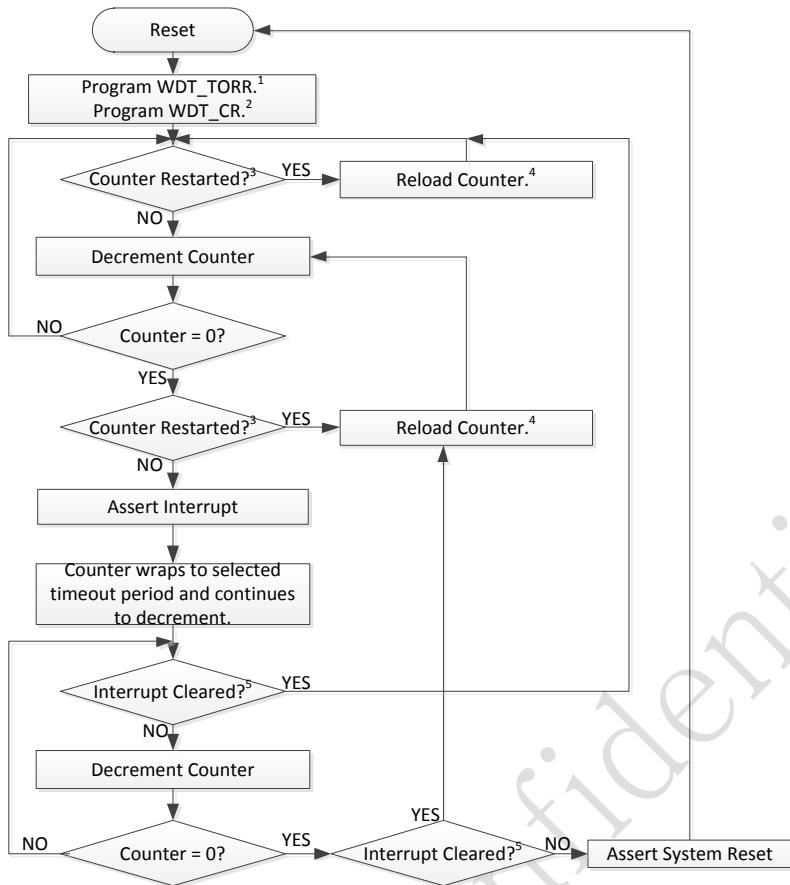
When a 0 is written to the output response mode field (RMOD, bit 1) of the Watchdog Timer Control Register (WDT\_CR), the WDT generates a system reset when a timeout occurs.

## **Reset Pulse Length**

The reset pulse length is the number of pclk cycles for which a system reset is asserted. When a system reset is generated, it remains asserted for the number of cycles specified by the reset pulse length or until the system is reset. A counter restart has no effect on the system reset once it has been asserted.

### **50.3.2 Programming sequence**

#### **Operation Flow Chart (Response mode=1)**



1. Select required timeout period.
2. Set reset pulse length, response mode, and enable WDT.
3. Write 0x76 to WDT\_CRR.
4. Starts back to selected timeout period.
5. Can clear by reading WDT\_EOI or restarting (kicking) the counter by writing 0x76 to WDT\_CRR.

Fig. 50-2 WDT Operation Flow

## 50.4 Register Description

This section describes the control/status registers of the design.

### 50.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
WDT_CR	0x0000	W	0x0000000a	Control Register
WDT_TORR	0x0004	W	0x00000000	Timeout range Register
WDT_CCVR	0x0008	W	0x00000000	Current counter value Register
WDT_CRR	0x000c	W	0x00000000	Counter restart Register
WDT_STAT	0x0010	W	0x00000000	Interrupt status Register
WDT_EOI	0x0014	W	0x00000000	Interrupt clear Register

Notes: Size: **B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**-WORD (32 bits) access

### 50.4.2 Detail Register Description

#### WDT\_CR

Address: Operational Base + offset (0x0000)

Control Register

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4:2	RW	0x2	<p>rst_pluse_lenth Reset pulse length. This is used to select the number of pclk cycles for which the system reset stays asserted.</p> <p>3'b000: 2 pclk cycles 3'b001: 4 pclk cycles 3'b010: 8 pclk cycles 3'b011: 16 pclk cycles 3'b100: 32 pclk cycles 3'b101: 64 pclk cycles 3'b110: 128 pclk cycles 3'b111: 256 pclk cycles</p>
1	RW	0x1	<p>resp_mode Response mode. Selects the output response generated to a timeout.</p> <p>1'b0: Generate a system reset 1'b1: First generate an interrupt and if it is not cleared by the time a second timeout occurs then generate a system reset</p>
0	RW	0x0	<p>wdt_en WDT enable</p> <p>1'b0: WDT disabled 1'b1: WDT enabled</p>

**WDT\_TORR**

Address: Operational Base + offset (0x0004)

Timeout range Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3:0	RW	0x0	<p>timeout_period Timeout period. This field is used to select the timeout period from which the watchdog counter restarts. A change of the timeout period takes effect only after the next counter restart (kick). The range of values available for a 32-bit watchdog counter are:</p> <p>4'b0000: 0x0000ffff 4'b0001: 0x0001ffff 4'b0010: 0x0003ffff 4'b0011: 0x0007ffff 4'b0100: 0x000fffff 4'b0101: 0x001fffff 4'b0110: 0x003fffff 4'b0111: 0x007fffff 4'b1000: 0x00ffffff 4'b1001: 0x01ffffff 4'b1010: 0x03ffffff 4'b1011: 0x07ffffff 4'b1100: 0x0fffffff 4'b1101: 0x1fffffff 4'b1110: 0x3fffffff 4'b1111: 0x7fffffff</p>

**WDT\_CCVR**

Address: Operational Base + offset (0x0008)

Current counter value Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	<p>cur_cnt Current counter value This register, when read, is the current value of the internal counter. This value is read coherently whenever it is read</p>

**WDT\_CRR**

Address: Operational Base + offset (0x000c)

Counter restart Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	W1C	0x00	<p>cnt_restart Counter restart</p> <p>This register is used to restart the WDT counter. As a safety feature to prevent accidental restarts, the value 0x76 must be written. A restart also clears the WDT interrupt. Reading this register returns zero.</p>

**WDT\_STAT**

Address: Operational Base + offset (0x0010)

Interrupt status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RO	0x0	<p>wdt_status</p> <p>This register shows the interrupt status of the WDT.</p> <p>1'b1: Interrupt is active regardless of polarity.</p> <p>1'b0: Interrupt is inactive.</p>

**WDT\_EOI**

Address: Operational Base + offset (0x0014)

Interrupt clear Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RO	0x0	<p>wdt_int_clr</p> <p>Clears the watchdog interrupt.</p> <p>This can be used to clear the interrupt without restarting the watchdog counter.</p>

**50.5 Application Notes**

Please refer to the function description section.

## Chapter 51 Pulse Width Modulation (PWM)

### 51.1 Overview

The pulse-width modulator (PWM) feature is very common in embedded systems. It provides a way to generate a pulse periodic waveform for motor control or can act as a digital-to-analog converter with some external components.

The PWM Module supports the following features:

- 4-built-in PWM channels
- Configurable to operate in capture mode
  - Measures the high/low polarity effective cycles of this input waveform
  - Generates a single interrupt at the transition of input waveform polarity
  - 32-bit high polarity capture register
  - 32-bit low polarity capture register
  - 32-bit current value register
- Configurable to operate in continuous mode or one-shot mode
  - 32-bit period counter
  - 32-bit duty register
  - 32-bit current value register
  - Configurable PWM output polarity in inactive state and duty period pulse polarity
  - Period and duty cycle are shadow buffered. Change takes effect when the end of the effective period is reached or when the channel is disabled
  - Programmable center or left aligned outputs, and change takes effect when the end of the effective period is reached or when the channel is disabled
  - 8-bit repeat counter for one-shot operation. One-shot operation will produce  $N + 1$  periods of the waveform, where  $N$  is the repeat counter value, and generates a single interrupt at the end of operation
  - Continuous mode generates the waveform continuously, and do not generates any interrupts
- pre-scaled operation to bus clock and then further scaled
- Available low-power mode to reduce power consumption when the channel is inactive.

## 51.2 Block Diagram

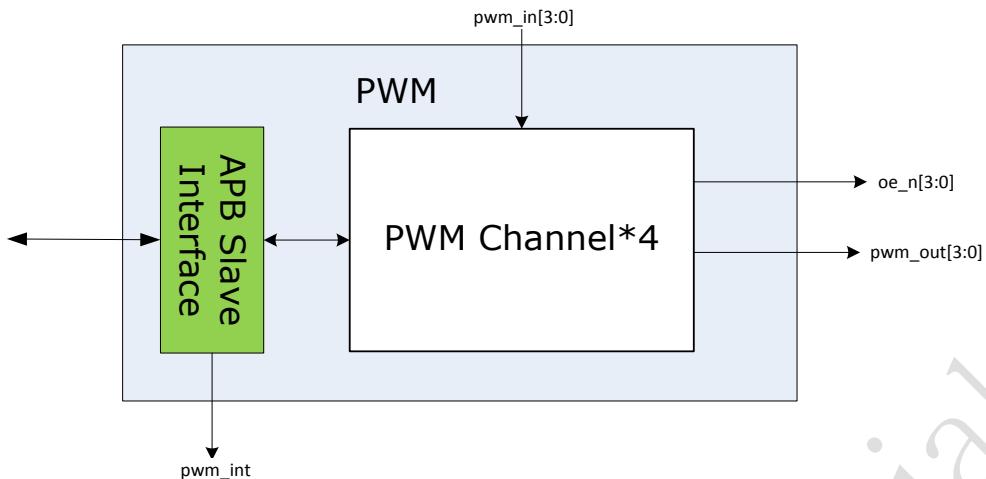


Fig. 51-1 PWM architecture

### 51.2.1 PWM APB Slave Interface

The host processor gets access to PWM Register Block through the APB slave interface with 32-bit bus width, and asserts the active-high level interrupt.

### 51.2.2 PWM Channels

This is the control logic of PWM module, and controls the operation of PWM module according to the configured working mode.

## 51.3 Functional description

The PWM supports three operation modes: reference mode, one-shot mode and continuous mode. For the one-shot mode and the continuous mode, the PWM output can be configured as the left-aligned mode or the center-aligned mode.

### 51.3.1 Reference mode

The reference mode is used to measure the PWM channel input waveform high/low effective cycles with the PWM channel clock, and asserts an interrupt when the polarity of the input waveform changes. The number of the high effective cycles is recorded in the PWMx\_PERIOD\_HPC register, while the number of the low effective cycles is recorded in the PWMx\_DUTY\_LPC register.

Note: the PWM input waveform is doubled buffered when the PWM channel is working in order to filter unexpected shot-time polarity transition, and therefore the interrupt is asserted several cycles after the input waveform polarity changes, and so does the change of the values of PWMx\_PERIOD\_HPC and PWMx\_DUTY\_LPC.

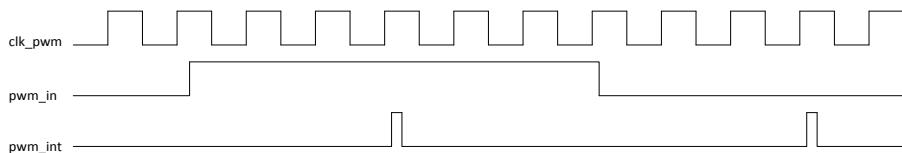


Fig. 51-2 PWM Reference Mode

### 51.3.2 Continuous Mode

The PWM channel generates a series of the pulses continuously as expected once the channel

is enabled with continuous mode.

In the continuous mode, the PWM output waveforms can be in one form of the two output mode: left-aligned mode or center-aligned mode.

For the left-aligned output mode, the PWM channel firstly starts the duty cycle with the configured duty polarity (PWMx\_CTRL.duty\_pol). Once duty cycle number (PWMx\_DUTY\_LPC) is reached, the output is switched to the opposite polarity. After the period number (PWMx\_PERIOD\_HPC) is reached, the output is again switched to the opposite polarity to start another period of desired pulse.

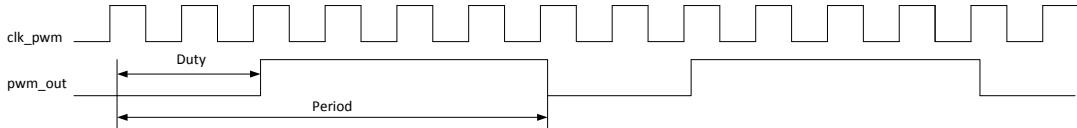


Fig. 51-3 PWM Left-aligned Output Mode

For the center-aligned output mode, the PWM channel firstly starts the duty cycle with the configured duty polarity (PWMx\_CTRL.duty\_pol). Once one half of duty cycle number (PWMx\_DUTY\_LPC) is reached, the output is switched to the opposite polarity. Then if there is one half of duty cycle left for the whole period , the output is again switched to the opposite polarity. Finally after the period number (PWMx\_PERIOD\_HPC) is reached, the output starts another period of desired pulse.

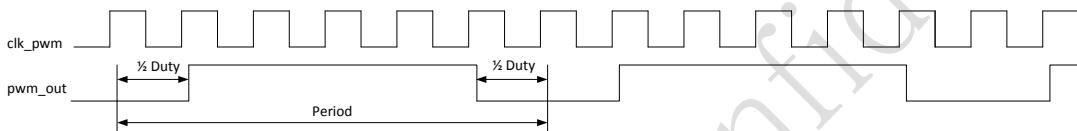


Fig. 51-4 PWM Center-aligned Output Mode

Once disable the PWM channel, the channel stops generating the output waveforms and output polarity is fixed as the configured inactive polarity (PWMx\_CTRL.inactive\_pol).

### 51.3.3 One-shot Mode

Unlike the continuous mode, the PWM channel generates the output waveforms within the configured periods (PWM\_CTRL.rpt + 1), and then stops. At the same times, an interrupt is asserted to inform that the operation has been finished.

There are also two output modes for the one-shot mode: the left-aligned mode and the center-aligned mode.

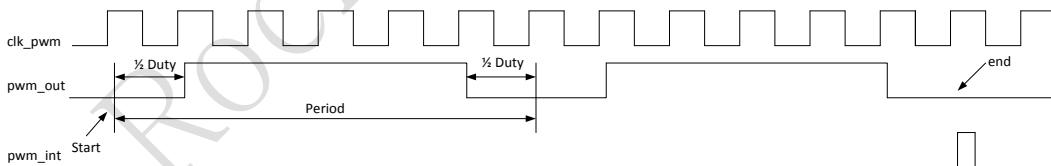


Fig. 51-5 PWM Center-aligned Output Mode

## 51.4 Register description

### 51.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
------	--------	------	-------------	-------------

Name	Offset	Size	Reset Value	Description
PWM_PWM0_CNT	0x0000	W	0x00000000	PWM Channel 0 Counter Register
PWM_PWM0_PERIOD_HPR	0x0004	W	0x00000000	PWM Channel 0 Period Register/High Polarity Capture Register
PWM_PWM0_DUTY_LPR	0x0008	W	0x00000000	PWM Channel 0 Duty Register/Low Polarity Capture Register
PWM_PWM0_CTRL	0x000c	W	0x00000000	PWM Channel 0 Control Register
PWM_PWM1_CNT	0x0010	W	0x00000000	PWM Channel 1 Counter Register
PWM_PWM1_PERIOD_HPR	0x0014	W	0x00000000	PWM Channel 1 Period Register/High Polarity Capture Register
PWM_PWM1_DUTY_LPR	0x0018	W	0x00000000	PWM Channel 1 Duty Register/Low Polarity Capture Register
PWM_PWM1_CTRL	0x001c	W	0x00000000	PWM Channel 1 Control Register
PWM_PWM2_CNT	0x0020	W	0x00000000	PWM Channel 2 Counter Register
PWM_PWM2_PERIOD_HPR	0x0024	W	0x00000000	PWM Channel 2 Period Register/High Polarity Capture Register
PWM_PWM2_DUTY_LPR	0x0028	W	0x00000000	PWM Channel 2 Duty Register/Low Polarity Capture Register
PWM_PWM2_CTRL	0x002c	W	0x00000000	PWM Channel 2 Control Register
PWM_PWM3_CNT	0x0030	W	0x00000000	PWM Channel 3 Counter Register
PWM_PWM3_PERIOD_HPR	0x0034	W	0x00000000	PWM Channel 3 Period Register/High Polarity Capture Register
PWM_PWM3_DUTY_LPR	0x0038	W	0x00000000	PWM Channel 3 Duty Register/Low Polarity Capture Register
PWM_PWM3_CTRL	0x003c	W	0x00000000	PWM Channel 3 Control Register
PWM_INTSTS	0x0040	W	0x00000000	Interrupt Status Register
PWM_INT_EN	0x0044	W	0x00000000	Interrupt Enable Register

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

## 51.4.2 Detail Register Description

### PWM\_PWM0\_CNT

Address: Operational Base + offset (0x0000)

PWM Channel 0 Counter Register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	CNT Timer Counter The 32-bit indicates current value of PWM Channel 0 counter. The counter runs at the rate of PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

### PWM\_PWM0\_PERIOD\_HPR

Address: Operational Base + offset (0x0004)

PWM Channel 0 Period Register/High Polarity Capture Register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	PERIOD_LPR Output Waveform Period/Input Waveform High Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the period of the output waveform. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the bit [31:1] is taken into account and bit [0] always considered as 0. If PWM is operated at the capture mode, this value indicates the effective high polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

### PWM\_PWM0\_DUTY\_LPR

Address: Operational Base + offset (0x0008)

PWM Channel 0 Duty Register/Low Polarity Capture Register

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	DUTY_LPR Output Waveform Duty Cycle/Input Waveform Low Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the duty cycle of the output waveform. The PWM starts the output waveform with duty cycle. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the [31:1] is taken into account. If PWM is operated at the capture mode, this value indicates the effective low polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

**PWM\_PWM0\_CTRL**

Address: Operational Base + offset (0x000c)

PWM Channel 0 Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	rpt Repeat Counter This field defines the repeated effective periods of output waveform in one-shot mode. The value N means N+1 repeated effective periods.
23:16	RW	0x00	scale Scale Factor This fields defines the scale factor applied to prescaled clock. The value N means the clock is divided by $2^N$ . If N is 0, it means that the clock is divided by 512( $2^{256}$ ).
15	RO	0x0	reserved
14:12	RW	0x0	prescale Prescale Factor This field defines the prescale factor applied to input clock. The value N means that the input clock is divided by $2^N$ .
11:10	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	clk_sel Clock Source Select 0: non-scaled clock is selected as PWM clock source. It means that the prescale clock is directly used as the PWM clock source 1: scaled clock is selected as PWM clock source
8	RW	0x0	lp_en Low Power Mode Enable 0: disabled 1: enabled When PWM channel is inactive state and Low Power Mode is enabled, the path to PWM Clock prescale module is blocked to reduce power consumption.
7:6	RO	0x0	reserved
5	RW	0x0	output_mode PWM Output mode 0: left aligned mode 1: center aligned mode
4	RW	0x0	inactive_pol Inactive State Output Polarity This defines the output waveform polarity when PWM channel is in inactive state. The inactive state means that PWM finishes the complete waveform in one-shot mode or PWM channel is disabled. 0: negative 1: positive
3	RW	0x0	duty_pol Duty Cycle Output Polarity This defines the polarity for duty cycle. PWM starts the output waveform with duty cycle. 0: negative 1: positive
2:1	RW	0x0	pwm_mode PWM Operation Mode 00: One shot mode. PWM produces the waveform within the repeated times defined by PWMx_CTRL_rpt . 01: Continuous mode. PWM produces the waveform continuously 10: Capture mode. PWM measures the cycles of high/low polarity of input waveform. 11: reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked the one-shot mode, this bit will be cleared at the end of operation

**PWM\_PWM1\_CNT**

Address: Operational Base + offset (0x0010)

PWM Channel 1 Counter Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	CNT Timer Counter The 32-bit indicates current value of PWM Channel 1 counter. The counter runs at the rate of PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

**PWM\_PWM1\_PERIOD\_HPR**

Address: Operational Base + offset (0x0014)

PWM Channel 1 Period Register/High Polarity Capture Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	PERIOD_LPR Output Waveform Period/Input Waveform High Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the period of the output waveform. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the bit [31:1] is taken into account and bit [0] always considered as 0. If PWM is operated at the capture mode, this value indicates the effective high polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

**PWM\_PWM1\_DUTY\_LPR**

Address: Operational Base + offset (0x0018)

PWM Channel 1 Duty Register/Low Polarity Capture Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	DUTY_LPR Output Waveform Duty Cycle/Input Waveform Low Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the duty cycle of the output waveform. The PWM starts the output waveform with duty cycle. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the [31:1] is taken into account. If PWM is operated at the capture mode, this value indicates the effective low polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

**PWM\_PWM1\_CTRL**

Address: Operational Base + offset (0x001c)

PWM Channel 1 Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	rpt Repeat Counter This field defines the repeated effective periods of output waveform in one-shot mode. The value N means N+1 repeated effective periods.
23:16	RW	0x00	scale Scale Factor This fields defines the scale factor applied to prescaled clock. The value N means the clock is divided by $2^N$ . If N is 0, it means that the clock is divided by 512( $2^{256}$ ).
15	RO	0x0	reserved
14:12	RW	0x0	prescale Prescale Factor This field defines the prescale factor applied to input clock. The value N means that the input clock is divided by $2^N$ .
11:10	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	clk_sel Clock Source Select 0: non-scaled clock is selected as PWM clock source. It means that the prescale clock is directly used as the PWM clock source 1: scaled clock is selected as PWM clock source
8	RW	0x0	lp_en Low Power Mode Enable 0: disabled 1: enabled When PWM channel is inactive state and Low Power Mode is enabled, the path to PWM Clock prescale module is blocked to reduce power consumption.
7:6	RO	0x0	reserved
5	RW	0x0	output_mode PWM Output mode 0: left aligned mode 1: center aligned mode
4	RW	0x0	inactive_pol Inactive State Output Polarity This defines the output waveform polarity when PWM channel is in inactive state. The inactive state means that PWM finishes the complete waveform in one-shot mode or PWM channel is disabled. 0: negative 1: positive
3	RW	0x0	duty_pol Duty Cycle Output Polarity This defines the polarity for duty cycle. PWM starts the output waveform with duty cycle. 0: negative 1: positive
2:1	RW	0x0	pwm_mode PWM Operation Mode 00: One shot mode. PWM produces the waveform within the repeated times defined by PWMx_CTRL_rpt 01: Continuous mode. PWM produces the waveform continuously 10: Capture mode. PWM measures the cycles of high/low polarity of input waveform. 11: reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked the one-shot mode, this bit will be cleared at the end of operation

**PWM\_PWM2\_CNT**

Address: Operational Base + offset (0x0020)

PWM Channel 2 Counter Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	CNT Timer Counter The 32-bit indicates current value of PWM Channel 2 counter. The counter runs at the rate of PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

**PWM\_PWM2\_PERIOD\_HPR**

Address: Operational Base + offset (0x0024)

PWM Channel 2 Period Register/High Polarity Capture Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	PERIOD_LPR Output Waveform Period/Input Waveform High Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the period of the output waveform. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the bit [31:1] is taken into account and bit [0] always considered as 0. If PWM is operated at the capture mode, this value indicates the effective high polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

**PWM\_PWM2\_DUTY\_LPR**

Address: Operational Base + offset (0x0028)

PWM Channel 2 Duty Register/Low Polarity Capture Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	DUTY_LPR Output Waveform Duty Cycle/Input Waveform Low Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the duty cycle of the output waveform. The PWM starts the output waveform with duty cycle. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the [31:1] is taken into account. If PWM is operated at the capture mode, this value indicates the effective low polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

**PWM\_PWM2\_CTRL**

Address: Operational Base + offset (0x002c)

PWM Channel 2 Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	rpt Repeat Counter This field defines the repeated effective periods of output waveform in one-shot mode. The value N means N+1 repeated effective periods.
23:16	RW	0x00	scale Scale Factor This fields defines the scale factor applied to prescaled clock. The value N means the clock is divided by $2^N$ . If N is 0, it means that the clock is divided by 512( $2^{256}$ ).
15	RO	0x0	reserved
14:12	RW	0x0	prescale Prescale Factor This field defines the prescale factor applied to input clock. The value N means that the input clock is divided by $2^N$ .
11:10	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	clk_sel Clock Source Select 0: non-scaled clock is selected as PWM clock source. It means that the prescale clock is directly used as the PWM clock source 1: scaled clock is selected as PWM clock source
8	RW	0x0	lp_en Low Power Mode Enable 0: disabled 1: enabled When PWM channel is inactive state and Low Power Mode is enabled, the path to PWM Clock prescale module is blocked to reduce power consumption.
7:6	RO	0x0	reserved
5	RW	0x0	output_mode PWM Output mode 0: left aligned mode 1: center aligned mode
4	RW	0x0	inactive_pol Inactive State Output Polarity This defines the output waveform polarity when PWM channel is in inactive state. The inactive state means that PWM finishes the complete waveform in one-shot mode or PWM channel is disabled. 0: negative 1: positive
3	RW	0x0	duty_pol Duty Cycle Output Polarity This defines the polarity for duty cycle. PWM starts the output waveform with duty cycle. 0: negative 1: positive
2:1	RW	0x0	pwm_mode PWM Operation Mode 00: One shot mode. PWM produces the waveform within the repeated times defined by PWMx_CTRL_rpt. 01: Continuous mode. PWM produces the waveform continuously 10: Capture mode. PWM measures the cycles of high/low polarity of input waveform. 11: reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked the one-shot mode, this bit will be cleared at the end of operation

**PWM\_PWM3\_CNT**

Address: Operational Base + offset (0x0030)

PWM Channel 3 Counter Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	CNT Timer Counter The 32-bit indicates current value of PWM Channel 3 counter. The counter runs at the rate of PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

**PWM\_PWM3\_PERIOD\_HPR**

Address: Operational Base + offset (0x0034)

PWM Channel 3 Period Register/High Polarity Capture Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	PERIOD_LPR Output Waveform Period/Input Waveform High Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the period of the output waveform. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the bit [31:1] is taken into account and bit [0] always considered as 0. If PWM is operated at the capture mode, this value indicates the effective high polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

**PWM\_PWM3\_DUTY\_LPR**

Address: Operational Base + offset (0x0038)

PWM Channel 3 Duty Register/Low Polarity Capture Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	DUTY_LPR Output Waveform Duty Cycle/Input Waveform Low Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the duty cycle of the output waveform. The PWM starts the output waveform with duty cycle. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the [31:1] is taken into account. If PWM is operated at the capture mode, this value indicates the effective low polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to ( $2^{32}-1$ ).

**PWM\_PWM3\_CTRL**

Address: Operational Base + offset (0x003c)

PWM Channel 3 Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:24	RW	0x00	rpt Repeat Counter This field defines the repeated effective periods of output waveform in one-shot mode. The value N means N+1 repeated effective periods.
23:16	RW	0x00	scale Scale Factor This fields defines the scale factor applied to prescaled clock. The value N means the clock is divided by $2^N$ . If N is 0, it means that the clock is divided by 512( $2^{256}$ ).
15	RO	0x0	reserved
14:12	RW	0x0	prescale Prescale Factor This field defines the prescale factor applied to input clock. The value N means that the input clock is divided by $2^N$ .
11:10	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
9	RW	0x0	clk_sel Clock Source Select 0: non-scaled clock is selected as PWM clock source. It means that the prescale clock is directly used as the PWM clock source 1: scaled clock is selected as PWM clock source
8	RW	0x0	lp_en Low Power Mode Enable 0: disabled 1: enabled When PWM channel is inactive state and Low Power Mode is enabled, the path to PWM Clock prescale module is blocked to reduce power consumption.
7:6	RO	0x0	reserved
5	RW	0x0	output_mode PWM Output mode 0: left aligned mode 1: center aligned mode
4	RW	0x0	inactive_pol Inactive State Output Polarity This defines the output waveform polarity when PWM channel is in inactive state. The inactive state means that PWM finishes the complete waveform in one-shot mode or PWM channel is disabled. 0: negative 1: positive
3	RW	0x0	duty_pol Duty Cycle Output Polarity This defines the polarity for duty cycle. PWM starts the output waveform with duty cycle. 0: negative 1: positive
2:1	RW	0x0	pwm_mode PWM Operation Mode 00: One shot mode. PWM produces the waveform within the repeated times defined by PWMx_CTRL_rpt 01: Continuous mode. PWM produces the waveform continuously 10: Capture mode. PWM measures the cycles of high/low polarity of input waveform. 11: reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked the one-shot mode, this bit will be cleared at the end of operation

**PWM\_INTSTS**

Address: Operational Base + offset (0x0040)

Interrupt Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:12	RO	0x0	reserved
11	RO	0x0	CH3_Pol Channel 3 Interrupt Polarity Flag This bit is used in capture mode in order to identify the transition of the input waveform when interrupt is generated. When bit is 1, please refer to PWM3_PERIOD_HPR to know the effective high cycle of Channel 0 input waveform. Otherwise, please refer to PWM3_PERIOD_HPR to know the effective low cycle of Channel 3 input waveform. Write 1 to CH3_IntSts will clear this bit.
10	RO	0x0	CH2_Pol Channel 2 Interrupt Polarity Flag This bit is used in capture mode in order to identify the transition of the input waveform when interrupt is generated. When bit is 1, please refer to PWM2_PERIOD_HPR to know the effective high cycle of Channel 0 input waveform. Otherwise, please refer to PWM2_PERIOD_HPR to know the effective low cycle of Channel 2 input waveform. Write 1 to CH2_IntSts will clear this bit.
9	RO	0x0	CH1_Pol Channel 1 Interrupt Polarity Flag This bit is used in capture mode in order to identify the transition of the input waveform when interrupt is generated. When bit is 1, please refer to PWM1_PERIOD_HPR to know the effective high cycle of Channel 0 input waveform. Otherwise, please refer to PWM1_PERIOD_HPR to know the effective low cycle of Channel 1 input waveform. Write 1 to CH1_IntSts will clear this bit.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
8	RO	0x0	CH0_Pol Channel 0 Interrupt Polarity Flag This bit is used in capture mode in order to identify the transition of the input waveform when interrupt is generated. When bit is 1, please refer to PWM0_PERIOD_HPR to know the effective high cycle of Channel 0 input waveform. Otherwise, please refer to PWM0_PERIOD_LPR to know the effective low cycle of Channel 0 input waveform. Write 1 to CH0_IntSts will clear this bit.
7:4	RO	0x0	reserved
3	RW	0x0	CH3_IntSts Channel 3 Interrupt Status 0: Channel 3 Interrupt not generated 1: Channel 3 Interrupt generated
2	RW	0x0	CH2_IntSts Channel 2 Interrupt Status 0: Channel 2 Interrupt not generated 1: Channel 2 Interrupt generated
1	RW	0x0	CH1_IntSts Channel 1 Interrupt Status 0: Channel 1 Interrupt not generated 1: Channel 1 Interrupt generated
0	RW	0x0	CH0_IntSts Channel 0 Raw Interrupt Status 0: Channel 0 Interrupt not generated 1: Channel 0 Interrupt generated

**PWM\_INT\_EN**

Address: Operational Base + offset (0x0044)

Interrupt Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:4	RO	0x0	reserved
3	RW	0x0	CH3_Int_en Channel 3 Interrupt Enable 0: Channel 3 Interrupt disabled 1: Channel 3 Interrupt enabled
2	RW	0x0	CH2_Int_en Channel 2 Interrupt Enable 0: Channel 2 Interrupt disabled 1: Channel 2 Interrupt enabled

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	CH1_Int_en Channel 1 Interrupt Enable 0: Channel 1 Interrupt disabled 1: Channel 1 Interrupt enabled
0	RW	0x0	CH0_Int_en Channel 0 Interrupt Enable 0: Channel 0 Interrupt disabled 1: Channel 0 Interrupt enabled

## 51.5 Interface Description

Table 51-1 PWM Interface Description

<b>Module pin</b>	<b>Direction</b>	<b>Pad name</b>	<b>IOMUX</b>
pwm3	O	GPIO7_C[7]	GPIO7CH_IOMUX[14:12]=011
pwm2	O	GPIO7_C[6]	GPIO7CH_IOMUX[9:8]=11
pwm1	O	GPIO7_A[1]	GPIO7A_IOMUX[2]=1
pwm0	O	GPIO7_A[0]	GPIO7A_IOMUX[1:0]=01

**51.6 Application**

## Notes

### 51.6.1 PWM Reference Mode Standard Usage Flow

1. Set PWMx\_CTRL.pwm\_en to '0' to disable the PWM channel.
2. Choose the prescale factor and the scale factor for pclk by programming PWMx\_CTRL.prescale and PWMx\_CTRL.scale, and select the clock needed by setting PWMx\_CTRL.clk\_sel.
3. Configure the channel to work in the reference mode.
4. Enable the INT\_EN.chx\_int\_en to enable the interrupt generation.
5. Enable the channel by writing '1' to PWMx\_CTRL.pwm\_en bit to start the channel.
6. When an interrupt is asserted, refer to INTSTS register to know the raw interrupt status. If the corresponding polarity flag is set, turn to PWMx\_PERIOD\_HPC register to know the effective high cycles of input waveforms, otherwise turn to PWMx\_DUTY\_LPC register to know the effective low cycles.
7. Write '0' to PWMx\_CTRL.pwm\_en to disable the channel.

### 51.6.2 PWM One-shot Mode/Continuous Standard Usage Flow

1. Set PWMx\_CTRL.pwm\_en to '0' to disable the PWM channel.
2. Choose the prescale factor and the scale factor for pclk by programming PWMx\_CTRL.prescale and PWMx\_CTRL.scale, and select the clock needed by setting PWMx\_CTRL.clk\_sel.
3. Choose the output mode by setting PWMx\_CTRL.output\_mode, and set the duty polarity and inactive polarity by programming PWMx\_CTRL.duty\_pol and PWMx\_CTRL.inactive\_pol.
4. Set the PWMx\_CTRL.rpt if the channel is desired to work in the one-shot mode;
5. Configure the channel to work in the one-shot mode or the continuous mode.

6. Enable the INT\_EN.chx\_int\_en to enable the interrupt generation if the channel is desired to work in the one-shot mode;
7. If the channel is working in the one-shot mode, an interrupt is asserted after the end of operation, and the PWMx\_CTRL.pwm\_en is automatically cleared. Whatever mode the channel is working, write '0' to PWMx\_CTRL.pwm\_en bit to disable the PWM channel.

### **51.6.3 Low-power mode**

Setting PWMx\_CTRL.lp\_en to '1' makes the channel enter the low-power mode. When the PWM channel is inactive, the APB bus clock to the clock prescale module is gated in order to reduce the power consumption. It is recommended to disable the channel before entering the low-power mode, and quit the low-power mode before enabling the channel.

### **51.6.4 Other notes**

When the channel is active to produce waveforms, it is free to programming the PWMx\_PERIOD\_HPC and PWMx\_DUTY\_LPC register. The change will not take effect immediately until the current period ends.

An active channel can be changed to another operation mode without disable the PWM channel. However, during the transition of the operation mode there may be some irregular output waveforms. So does changing the clock division factor when the channel is active.

In RK3288, user need to configure the SOC\_CON2[0] to "1'b1" before using the PWM.

## Chapter 52 I2C Interface

### 52.1 Overview

The Inter-Integrated Circuit (I2C) is a two wired (SCL and SDA), bi-directional serial bus that provides an efficient and simple method of information exchange between devices. This I2C bus controller supports master mode acting as a bridge between AMBA protocol and generic I2C bus system.

I2C Controller supports the following features:

- Item Compatible with I2C-bus
- AMBA APB slave interface
- Supports master mode of I2C bus
- Software programmable clock frequency and transfer rate up to 400Kbit/sec
- Supports 7 bits and 10 bits addressing modes
- Interrupt or polling driven multiple bytes data transfer
- Clock stretching and wait state generation
- There are two I2C controller in bus sub-system:I2C PMU, and I2C AUDIO. There are four I2C controller in peripheral sub-system: I2C SENSOR, I2C CAM, I2C\_TP, and I2C\_HDMI.

### 52.2 Block Diagram

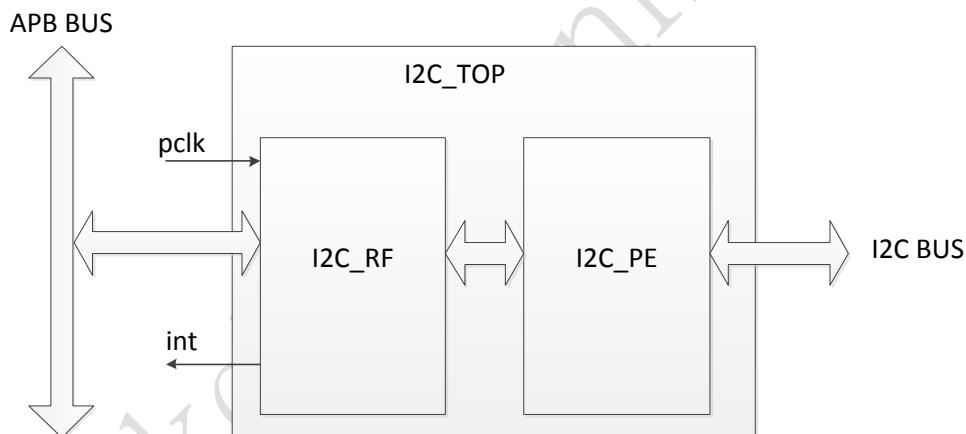


Fig. 52-1 I2C architecture

#### I2C\_RF

I2C\_RF module is used to control the I2C controller operation by the host with APB interface. It implements the register set and the interrupt functionality. The CSR component operates synchronously with the pclk clock.

#### I2C\_PE

I2C\_PE module implements the I2C master operation for transmit data to and receive data from other I2C devices. The I2C master controller operates synchronously with the pclk.

#### I2C\_TOP

I2C\_TOP module is the top module of the I2C controller.

### 52.3 Function description

This chapter provides a description about the functions and behavior under various conditions.

The I2C controller supports only Master function. It supports the 7-bits/10-bits addressing mode and support general call address. The maximum clock frequency and transfer rate can be up to 400Kbit/sec.

The operations of I2C controller is divided to 2 parts and described separately: initialization and master mode programming.

### **52.3.1 Initialization**

The I2C controller is based on AMBA APB bus architecture and usually is part of a SOC. So before I2C operates, some system setting & configuration must be conformed, which includes:

- I2C interrupt connection type: CPU interrupt scheme should be considered. If the I2C interrupt is connected to extra Interrupt Controller module, we need decide the INTC vector.
- I2C Clock Rate: The I2C controller uses the APB clock as the system clock so the APB clock will determine the I2C bus clock. The correct register setting is subject to the system requirement.

### **52.3.2 Master Mode Programming**

**1. SCL Clock:** When the I2C controller is programmed in Master mode, the SCL frequency is determined by I2C\_CLKDIV register. The SCL frequency is calculated by the following formula:

$$\text{SCL Divisor} = 8 * (\text{CLKDIVL} + 1 + \text{CLKDIVH} + 1)$$

$$\text{SCL} = \text{PCLK} / \text{SCLK Divisor}$$

#### **2. Data Receiver Register Access**

When the i2c controller received MRXCNT bytes data, CPU can get the datas through register RXDATA0 ~ RXDATA7. The controller can receive up to 32 byte data in one transaction.

When MRXCNT register is written, the I2C controller will start to drive SCL to receive data.

#### **3. Transmit Trasmitter Register**

Data to transmit are written to TXDATA0~7 by CPU. The controller can transmit up to 32 byte data in one transaction. The lower byte will be transmitted first.

When MTXCNT register is written, the I2C controller will start to transmit data.

#### **4. Start Command**

Write 1 to I2C\_CON[3], the controller will send I2C start command.

#### **5. Stop Command**

Write 1 to I2C\_CON[4], the controller will send I2C stop command

#### **6. I2C Operation mode**

There are four i2c operation modes.

When I2C\_CON[2:1] is 2'b00, the controller transmit all valid data in TXDATA0~TXDATA7 byte by byte. The controller will transmit lower byte first.

When I2C\_CON[2:1] is 2'b01, the controller will transmit device address in MRXADDR first (Write/Read bit = 0) and then transmit device register address in MRXRADDR. After that, the controller will assert restart signal and resend MRXADDR (Write/Read bit = 1). At last, the controller enter receive mode.

When I2C\_CON[2:1] is 2'b10, the controller is in receive mode, it will triggered clock to read MRXCNT byte data.

When I2C\_CON[2:1] is 2'b11, the controller will transmit device address in MRXADDR first (Write/Read bit = 1) and then transmit device register address in MRXRADDR . After that, the controller will assert restart signal and resend MRXADDR (Write/Read bit = 1). At last, the controller enter receive mode.

## **7. Read/Write Command**

When I2C\_OPMODE(I2C\_CON[2:1]) is 2'b01 or 2'b11, the Read/Write command bit is decided by controller itself.

In RX only mode (I2C\_CON[2:1] is 2'b10), the Read/Write command bit is decided by MRXADDR[0].

In TX only mode (I2C\_CON[[2:1] is 2'b00), the Read/Write command bit is decided by TXDATA[0].

## **8. Master Interrupt Condition**

There are 7 interrupt bits in I2C\_ISR register related to master mode.

Byte transmitted finish interrupt (Bit 0): The bit is asserted when Master complete transmitting a byte.

Byte received finish interrupt (Bit 1): The bit is asserted when Master complete receiving a byte.

MTXCNT bytes data transmitted finish interrupt (Bit 2): The bit is asserted when Master complete transmitting MTXCNT bytes.

MRXCNT bytes data received finish interrupt (Bit 3): The bit is asserted when Master complete receiving MRXCNT bytes.

Start interrupt (Bit 4): The bit is asserted when Master finish asserting start command to I2C bus.

Stop interrupt (Bit 5): The bit is asserted when Master finish asserting stop command to I2C bus.

NAK received interrupt (Bit 6): The bit is asserted when Master received a NAK handshake.

## **9. Last byte acknowledge control**

If I2C\_CON[5] is 1, the I2C controller will transmit NAK handshake to slave when the last byte received in RX only mode.

If I2C\_CON[5] is 0, the I2C controller will transmit ACK handshake to slave when the last byte received in RX only mode.

## **10. How to handle nak handshake received**

If I2C\_CON[6] is 1, the I2C controller will stop all transactions when NAK handshake received. And the software should take responsibility to handle the problem.

If I2C\_CON[6] is 0, the I2C controller will ignore all NAK handshake received.

## **11. I2C controller data transfer waveform**

### **● Bit transferring**

#### **(a) Data Validity**

The SDA line must be stable during the high period of SCL, and the data on SDA line can only be changed when SCL is in low state.

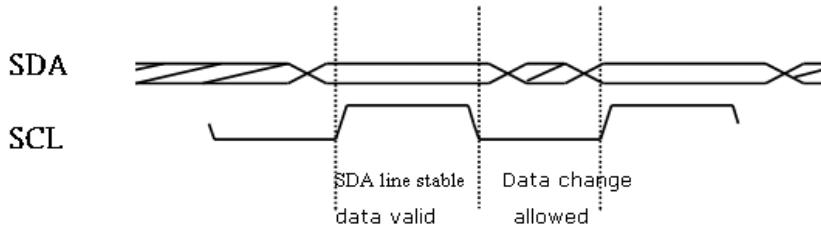


Fig. 52-2 I2C DATA Validity

## (b) START and STOP conditions

START condition occurs when SDA goes low while SCL is in high period. STOP condition is generated when SDA line goes high while SCL is in high state.

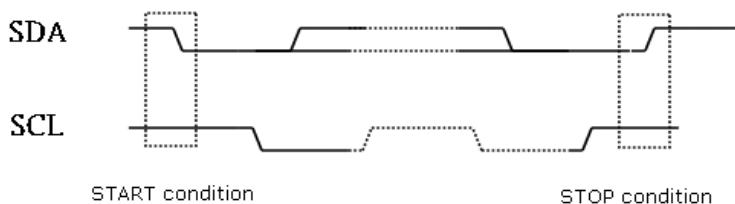


Fig. 52-3 I2C Start and stop conditions

## ● Data transfer

## (a) Acknowledge

After a byte of data transferring (clocks labeled as 1~8), in 9<sup>th</sup> clock the receiver must assert an ACK signal on SDA line, if the receiver pulls SDA line to low, it means "ACK", on the contrary, it's "NOT ACK".



Fig. 52-4 I2C Acknowledge

## (b) Byte transfer

The master own I2C bus might initiate multi byte to transfer to a slave. The transfer starts from a "START" command and ends in a "STOP" command. After every byte transfer, the receiver must reply an ACK to transmitter.

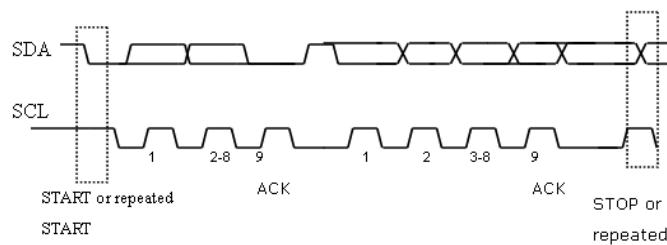


Fig. 52-5 I2C byte transfer

## 52.4 Register Description

This section describes the control/status registers of the design.

## 52.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
I2C_CON	0x0000	W	0x00000000	control register
I2C_CLKDIV	0x0004	W	0x00060006	Clock divisor register
I2C_MRXXADDR	0x0008	W	0x00000000	the slave address accessed for master receive mode
I2C_MRXRADDR	0x000c	W	0x00000000	the slave register address accessed for master receive mode
I2C_MTXCNT	0x0010	W	0x00000000	master transmit count
I2C_MRXCNT	0x0014	W	0x00000000	master receive count
I2C_IEN	0x0018	W	0x00000000	interrupt enable register
I2C_IPD	0x001c	W	0x00000000	interrupt pending register
I2C_FCNT	0x0020	W	0x00000000	finished count
I2C_TXDATA0	0x0100	W	0x00000000	I2C transmit data register 0
I2C_TXDATA1	0x0104	W	0x00000000	I2C transmit data register 1
I2C_TXDATA2	0x0108	W	0x00000000	I2C transmit data register 2
I2C_TXDATA3	0x010c	W	0x00000000	I2C transmit data register 3
I2C_TXDATA4	0x0110	W	0x00000000	I2C transmit data register 4
I2C_TXDATA5	0x0114	W	0x00000000	I2C transmit data register 5
I2C_TXDATA6	0x0118	W	0x00000000	I2C transmit data register 6
I2C_TXDATA7	0x011c	W	0x00000000	I2C transmit data register 7
I2C_RXDATA0	0x0200	W	0x00000000	I2C receive data register 0
I2C_RXDATA1	0x0204	W	0x00000000	I2C receive data register 1
I2C_RXDATA2	0x0208	W	0x00000000	I2C receive data register 2
I2C_RXDATA3	0x020c	W	0x00000000	I2C receive data register 3
I2C_RXDATA4	0x0210	W	0x00000000	I2C receive data register 4
I2C_RXDATA5	0x0214	W	0x00000000	I2C receive data register 5
I2C_RXDATA6	0x0218	W	0x00000000	I2C receive data register 6
I2C_RXDATA7	0x021c	W	0x00000000	I2C receive data register 7

Notes: **Size:** **B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**-WORD (32 bits) access

## 52.4.2 Detail Register Description

### I2C\_CON

Address: Operational Base + offset (0x0000)  
control register

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	act2nak operation when NAK handshake is received 1'b0: ignored 1'b1: stop transaction

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	ack last byte acknowledge control last byte acknowledge control in master receive mode . 1'b0: ACK 1'b1: NAK
4	W1C	0x0	stop stop enable when this bit is written to 1, I2C will generate stop signal. It cleared itself when stop operation ends.
3	W1C	0x0	start start enable when this bit is written to 1, I2C will generate start signal. It cleared itself when start operation ends.
2:1	RW	0x0	i2c_mode 2'b00: transmit only 2'b01: transmit address (device + register address) --> restart --> transmit address -> receive only 2'b10: receive only 2'b11: transmit address (device + register address, write/read bit is 1) --> restart --> transmit address (device address) --> receive data
0	RW	0x0	i2c_en i2c module enable

**I2C\_CLKDIV**

Address: Operational Base + offset (0x0004)

Clock divisor register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:16	RW	0x0006	CLKDIVH SCL high level clock count $T(SCL\_HIGH) = T(PCLK) * CLKDIVH * 8$
15:0	RW	0x0006	CLKDIVL SCL low level clock count $T(SCL\_LOW) = T(PCLK) * CLKDIVL * 8$

**I2C\_MRXADDR**

Address: Operational Base + offset (0x0008)

the slave address accessed for master receive mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:27	RO	0x0	reserved
26	RW	0x0	addhvld address high byte valid
25	RW	0x0	addmvld address middle byte valid

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
24	RW	0x0	addlvid address low byte valid
23:0	RW	0x000000	saddr master address register the lowest bit indicate write or read

**I2C\_MRXRADDR**

Address: Operational Base + offset (0x000c)

the slave register address accessed for master receive mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:27	RO	0x0	reserved
26	RW	0x0	sraddhvld address high byte valid
25	RW	0x0	sraddmvld address middle byte valid
24	RW	0x0	sraddlvid address low byte valid
23:0	RW	0x000000	saddr slave register address accessed

**I2C\_MTXCNT**

Address: Operational Base + offset (0x0010)

master transmit count

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RW	0x00	mtxcnt master transmit count

**I2C\_MRXCNT**

Address: Operational Base + offset (0x0014)

master receive count

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RW	0x00	mrxcnt master receive count

**I2C\_IEN**

Address: Operational Base + offset (0x0018)

interrupt enable register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6	RW	0x0	nakrcvien NAK handshake received interrupt enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RW	0x0	stopien stop operation finished interrupt enable
4	RW	0x0	startien start operation finished interrupt enable
3	RW	0x0	mbrfien MRXCNT data received finished interrupt enable
2	RW	0x0	mbtfien MTXCNT data transmit finished interrupt enable
1	RW	0x0	brfien byte receive finished interrupt enable
0	RW	0x0	btfien byte transmit finished interrupt enable

**I2C\_IPD**

Address: Operational Base + offset (0x001c)

interrupt pending register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6	RW	0x0	nakrcvpd NAK handshake received interrupt pending bit
5	RW	0x0	stopipd stop operation finished interrupt pending bit
4	RW	0x0	startipd start operation finished interrupt pending bit
3	RW	0x0	mbrfpd MRXCNT data received finished interrupt pending bit
2	RW	0x0	mbtfpd MTXCNT data transmit finished interrupt pending bit
1	RW	0x0	brfpd byte receive finished interrupt pending bit
0	RW	0x0	btfpd byte transmit finished interrupt pending bit

**I2C\_FCNT**

Address: Operational Base + offset (0x0020)

finished count

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:6	RO	0x0	reserved
5:0	RW	0x00	fcnt finished count the count of data which has been transmitted or received for debug purpose

**I2C\_TXDATA0**

Address: Operational Base + offset (0x0100)

I2C transmit data register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txdata0

**I2C\_TXDATA1**

Address: Operational Base + offset (0x0104)

I2C transmit data register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txdata1

**I2C\_TXDATA2**

Address: Operational Base + offset (0x0108)

I2C transmit data register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txdata2

**I2C\_TXDATA3**

Address: Operational Base + offset (0x010c)

I2C transmit data register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txdata3

**I2C\_TXDATA4**

Address: Operational Base + offset (0x0110)

I2C transmit data register 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txdata4

**I2C\_TXDATA5**

Address: Operational Base + offset (0x0114)

I2C transmit data register 5

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txdata5

**I2C\_TXDATA6**

Address: Operational Base + offset (0x0118)

I2C transmit data register 6

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txdata6

**I2C\_TXDATA7**

Address: Operational Base + offset (0x011c)

I2C transmit data register 7

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txdata7

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	txdata7

**I2C\_RXDATA0**

Address: Operational Base + offset (0x0200)

I2C receive data register 0

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	rxdata0

**I2C\_RXDATA1**

Address: Operational Base + offset (0x0204)

I2C receive data register 1

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	rxdata1

**I2C\_RXDATA2**

Address: Operational Base + offset (0x0208)

I2C receive data register 2

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	rxdata2

**I2C\_RXDATA3**

Address: Operational Base + offset (0x020c)

I2C receive data register 3

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	rxdata3

**I2C\_RXDATA4**

Address: Operational Base + offset (0x0210)

I2C receive data register 4

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	rxdata4

**I2C\_RXDATA5**

Address: Operational Base + offset (0x0214)

I2C receive data register 5

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	rxdata5

**I2C\_RXDATA6**

Address: Operational Base + offset (0x0218)

I2C receive data register 6

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	rxdata6

**I2C\_RXDATA7**

Address: Operational Base + offset (0x021c)

I2C receive data register 7

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	rxdata7

**52.5 Interface description**

Table 52-1 I2C Interface Description

<b>Module pin</b>	<b>Direction</b>	<b>Pad name</b>	<b>IOMUX</b>
I2C PMU Interface			
i2c_pmu_sda	I/O	IO_I2C0PMUsda_P MUGPIO0b7	GRF_GPIO0B_IOMUX[1:4]=1
i2c_pmu_scl	I/O	IO_I2C0PMUscl_P MUGPIO0c0	GRF_GPIO0C_IOMUX[0]=1
I2C SENSOR Interface			
i2c_sensor_sda	I/O	IO_I2C1SENSORsd a_SCrst_GPIO1830 gpio8a4	GRF_GPIO8A_IOMUX[9:8]=01
i2c_sensor_scl	I/O	IO_I2C1SENSORsc l_SCclk_GPIO1830 gpio8a5	GRF_GPIO8A_IOMUX[1:10]=01
I2C AUDIO Interface			
i2c_audio_sda	I/O	IO_I2C2AUDIOsda _AUDIOGPIO6b1	GRF_GPIO6B_IOMUX[2]=1
i2c_audio_scl	I/O	IO_I2C2AUDIOscl _AUDIOGPIO6b2	GRF_GPIO6B_IOMUX[4]=1
I2C CAM Interface			
i2c_cam_sda	I/O	IO_I2C3CAMsda_D VPGPIO2c1	GRF_GPIO2C_IOMUX[2]=1
i2c_cam_scl	I/O	IO_I2C3CAMscl_D VPGPIO2c0	GRF_GPIO2C_IOMUX[0]=1
I2C TP Interface			
i2c_tp_sda	I/O	IO_I2C4TPsda_GPI O30GPIO7c1	GRF_GPIO7CL_IOMUX[4]=1
i2c_tp_scl	I/O	IO_I2C4TPscl_GPI O30GPIO7c2	GRF_GPIO7CL_IOMUX[8]=1
I2C HDMI Interface			
i2c_hdmi_sda	I/O	IO_I2C5HDMIsda _EDPHDMI2Csda_G PIO30GPIO7c3	GRF_GPIO7CL_IOMUX[13:12]=01
i2c_hdmi_scl	I/O	IO_I2C5HDMIscl_E DPHDMI2Cscl_GPI O30GPIO7c4	GRF_GPIO7CH_IOMUX[1:0]=01

and mix mode. Users are strongly advised to follow.

- Transmit only mode (I2C\_CON[1:0]=2'b00)

## 52.6 Applications

### Notes

The I2C controller core operation flow chart below is to describe how the software configures and performs an I2C transaction through this I2C controller core. Descriptions are divided into 3 sections, transmit only mode, receive only mode,

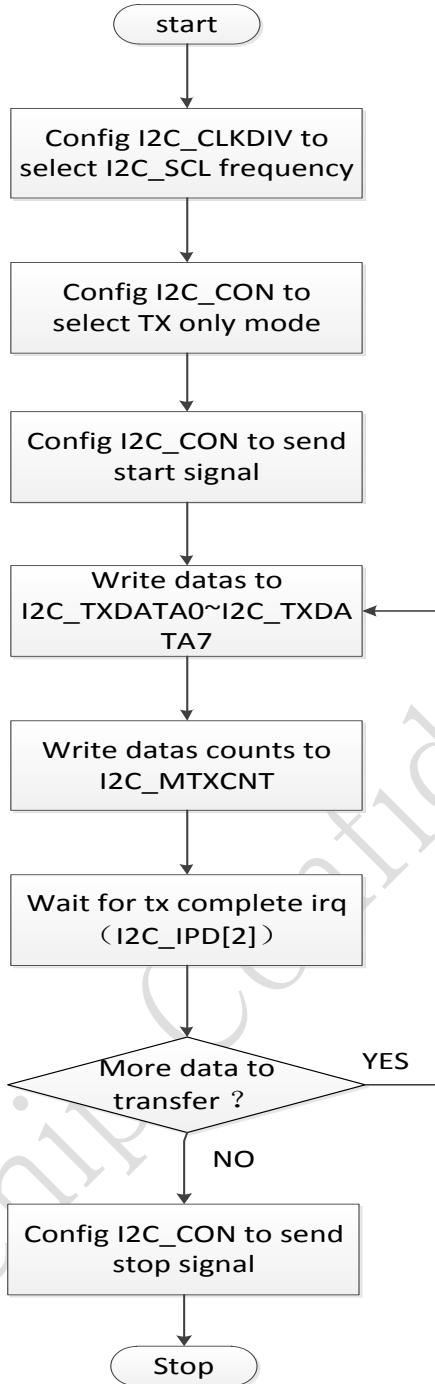


Fig. 52-6 I2C Flow chat for transmit only mode

- Receive only mode (I2C\_CON[1:0]=2'b10)

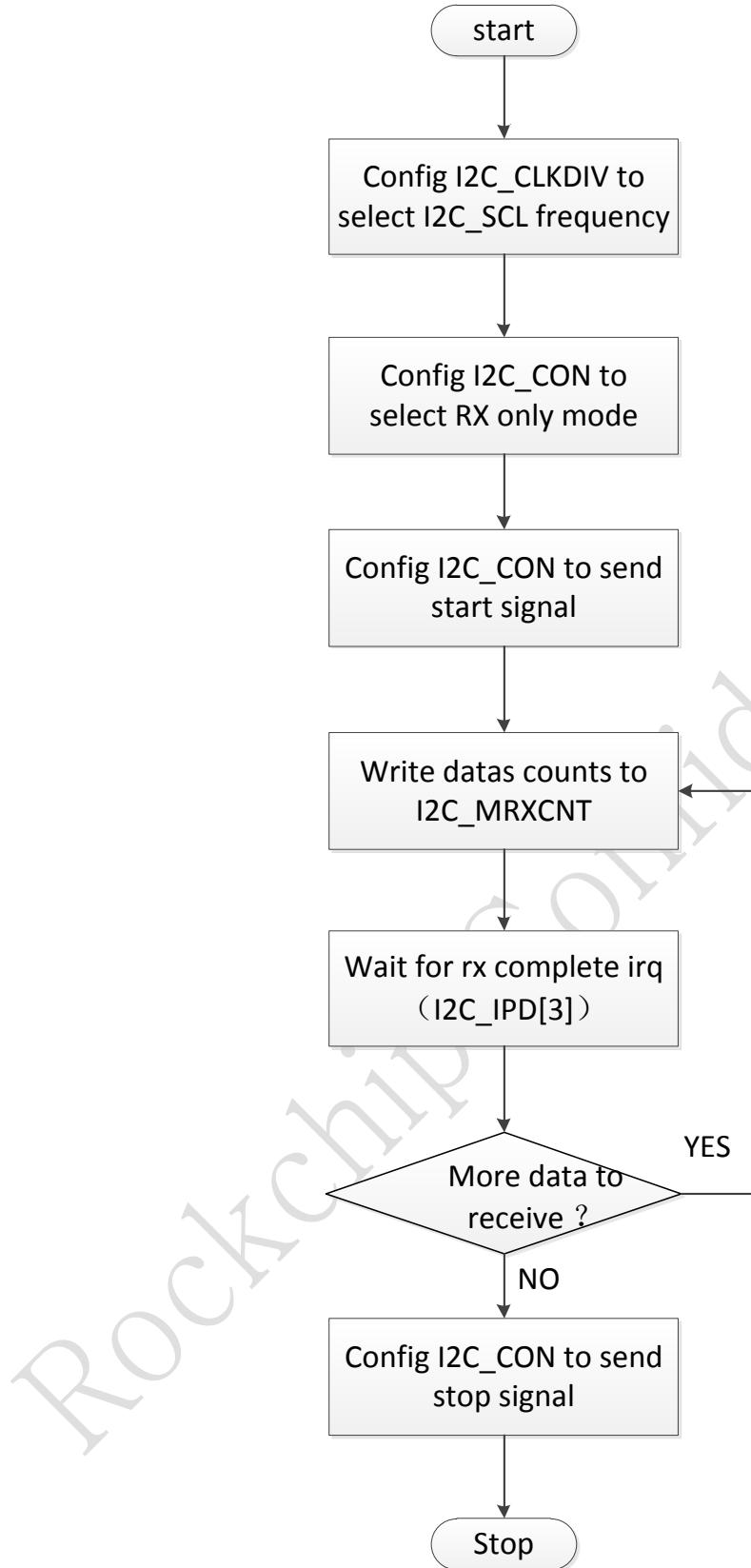


Fig. 52-7 I2C Flow chat for receive only mode

- Mix mode ( $I2C\_CON[1:0]=2'b01$  or  $I2C\_CON[1:0]=2'b11$ )

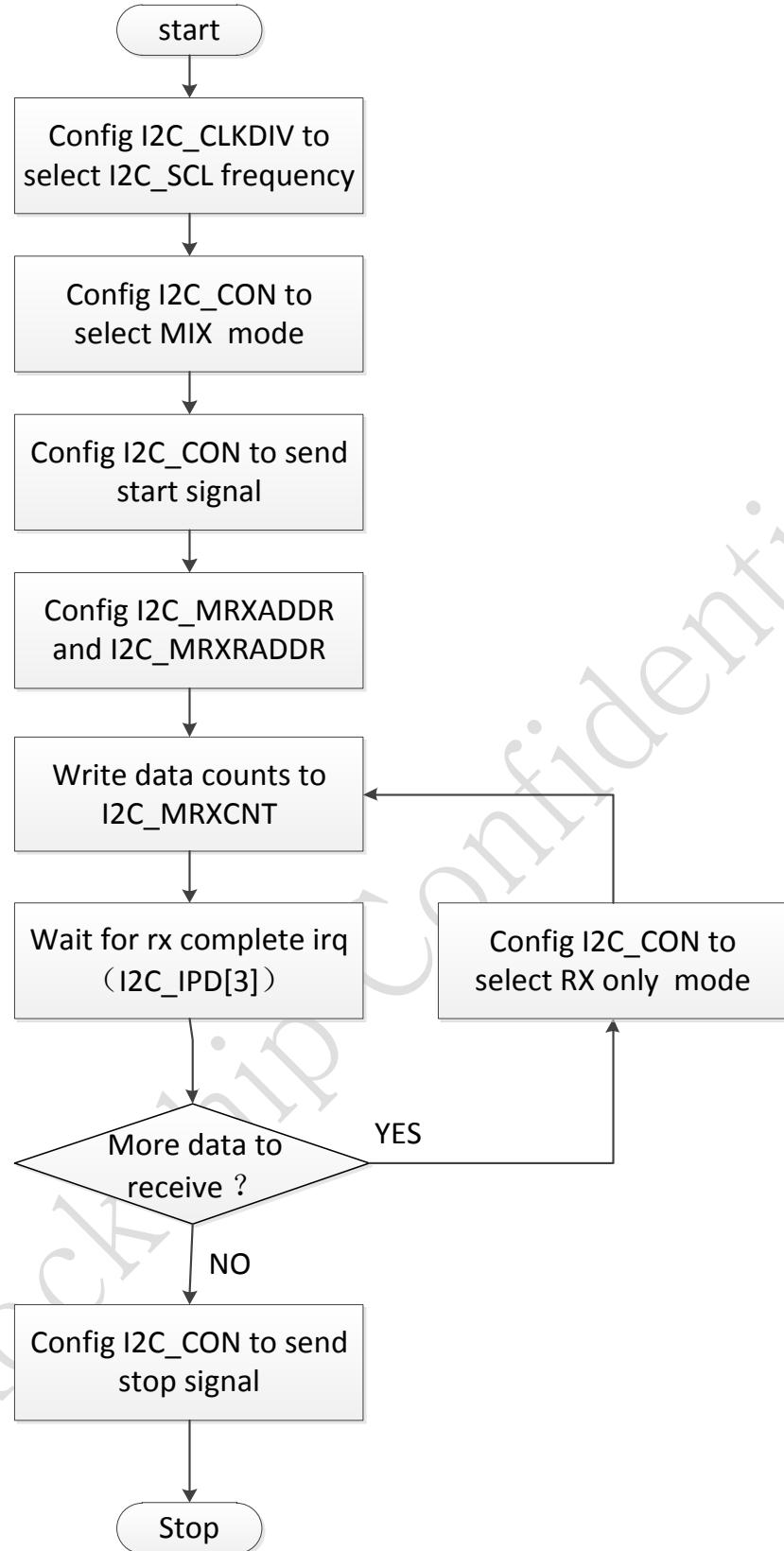


Fig. 52-8 I2C Flow chat for mix mode

# Chapter 53 Universal Asynchronous Receiver/Transmitter (UART)

## 53.1 Overview

The Universal Asynchronous Receiver/Transmitter (UART) is used for serial communication with a peripheral, modem (data carrier equipment, DCE) or data set. Data is written from a master (CPU) over the APB bus to the UART and it is converted to serial form and transmitted to the destination device. Serial data is also received by the UART and stored for the master (CPU) to read back.

UART Controller supports the following features:

- AMBA APB interface – Allows for easy integration into a Synthesizable Components for AMBA 2 implementation
- Support interrupt interface to interrupt controller
- Contain two 64Bytes FIFOs for data receive and transmit
- Programmable serial data baud rate as calculated by the following: baud rate = (serial clock frequency)/(16×divisor)
- UART\_BB/UART\_BT/UART\_GPS/UART\_EXP support auto flow-control, UART\_DBG do not support auto flow-control
- UART\_DBG support IrDA 1.0 SIR mode with up to 115.2 Kbaud data rate
- UART\_BB/UART\_BT/UART\_GPS/UART\_EXP are in peripheral subsystem, UART\_DBG is in bus subsystem

## 53.2 Block Diagram

This section provides a description about the functions and behavior under various conditions. The UART Controller comprises with:

- AMBA APB interface
- FIFO controllers
- Register block
- Modem synchronization block and baud clock generation block
- Serial receiver and serial transmitter

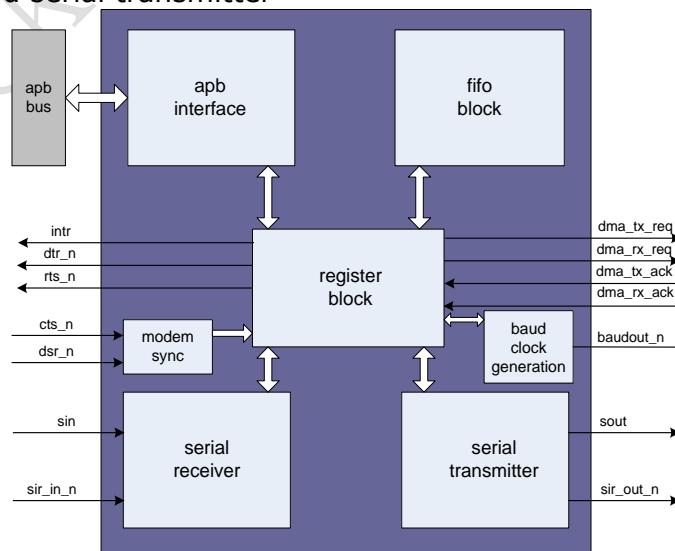


Fig. 53-1 UART Architecture

### APB INTERFACE

The host processor accesses data, control, and status information on the UART through the APB interface. The UART supports APB data bus widths of 8, 16, and 32 bits.

### **Register block**

Be responsible for the main UART functionality including control, status and interrupt generation.

### **Modem Synchronization block**

Synchronizes the modem input signal.

### **FIFO block**

Be responsible for FIFO control and storage (when using internal RAM) or signaling to control external RAM (when used).

### **Baud Clock Generator**

Generate the transmitter and receiver baud clock along with the output reference clock signal (baudout\_n).

### **Serial Transmitter**

Converts the parallel data, written to the UART, into serial form and adds all additional bits, as specified by the control register, for transmission. This makeup of serial data, referred to as a character can exit the block in two forms, either serial UART format or IrDA 1.0 SIR format.

### **Serial Receiver**

Converts the serial data character (as specified by the control register) received in either the UART or IrDA 1.0 SIR format to parallel form. Parity error detection, framing error detection and line break detection is carried out in this block.

## **53.3 Function description**

### **UART (RS232) Serial Protocol**

Because the serial communication is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end. An additional parity bit may be added to the serial character. This bit appears after the last data bit and before the stop bit(s) in the character structure to perform simple error checking on the received data, as shown in Figure.

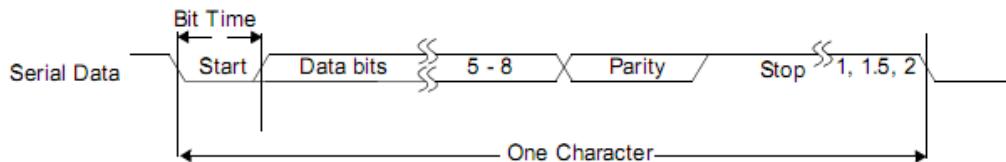


Fig. 53-2 UART Serial protocol

### **IrDA 1.0 SIR Protocol**

The Infrared Data Association (IrDA) 1.0 Serial Infrared (SIR) mode supports bi-directional data communications with remote devices using infrared radiation as the transmission medium. IrDA 1.0 SIR mode specifies a maximum baud rate of 115.2 Kbaud.

Transmitting a single infrared pulse signals a logic zero, while a logic one is represented by not sending a pulse. The width of each pulse is 3/16ths of a normal serial bit time. Data transfers can only occur in half-duplex fashion when IrDA SIR mode is enabled.

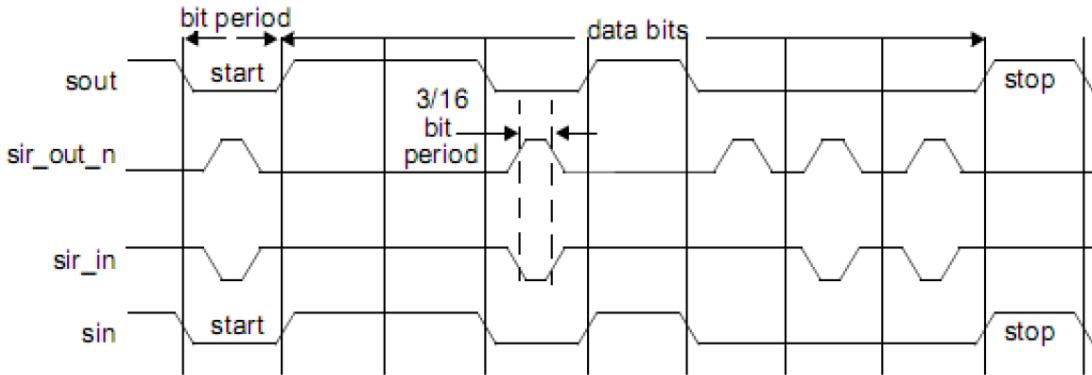


Fig. 53-3 IrDA 1.0

## Baud Clock

The baud rate is controlled by the serial clock (sclk or pclk in a single clock implementation) and the Divisor Latch Register (DLH and DLL). As the exact number of baud clocks that each bit was transmitted for is known, calculating the mid-point for sampling is not difficult, that is every 16 baud clocks after the mid point sample of the start bit.

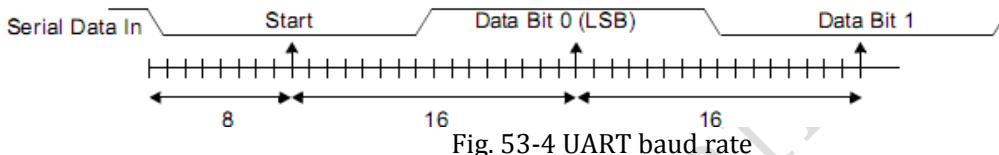


Fig. 53-4 UART baud rate

## FIFO Support

### 1. NONE FIFO MODE

If FIFO support is not selected, then no FIFOs are implemented and only a single receive data byte and transmit data byte can be stored at a time in the RBR and THR.

### 2. FIFO MODE

The FIFO depth of UART1/UART2/UART3 is 32bytes and the FIFO depth of UART0 is 64bytes. The FIFO mode of all the UART is enabled by register FCR[0].

## Interrupts

The following interrupt types can be enabled with the IER register.

- Receiver Error
- Receiver Data Available
- Character Timeout (in FIFO mode only)
- Transmitter Holding Register Empty at/below threshold (in Programmable THRE Interrupt mode)
- Modem Status

## DMA Support

The uart supports DMA signaling with the use of two output signals (dma\_tx\_req\_n and dma\_rx\_req\_n) to indicate when data is ready to be read or when the transmit FIFO is empty.

The dma\_tx\_req\_n signal is asserted under the following conditions:

- When the Transmitter Holding Register is empty in non-FIFO mode.
- When the transmitter FIFO is empty in FIFO mode with Programmable THRE interrupt mode disabled.
- When the transmitter FIFO is at, or below the programmed threshold with Programmable THRE interrupt mode enabled.

The `dma_rx_req_n` signal is asserted under the following conditions:

- When there is a single character available in the Receive Buffer Register in non-FIFO mode.
- When the Receiver FIFO is at or above the programmed trigger level in FIFO mode.

### Auto Flow Control

The UART can be configured to have a 16750-compatible Auto RTS and Auto CTS serial data flow control mode available. If FIFOs are not implemented, then this mode cannot be selected. When Auto Flow Control mode has been selected, it can be enabled with the Modem Control Register (MCR[5]). Following figure shows a block diagram of the Auto Flow Control functionality.

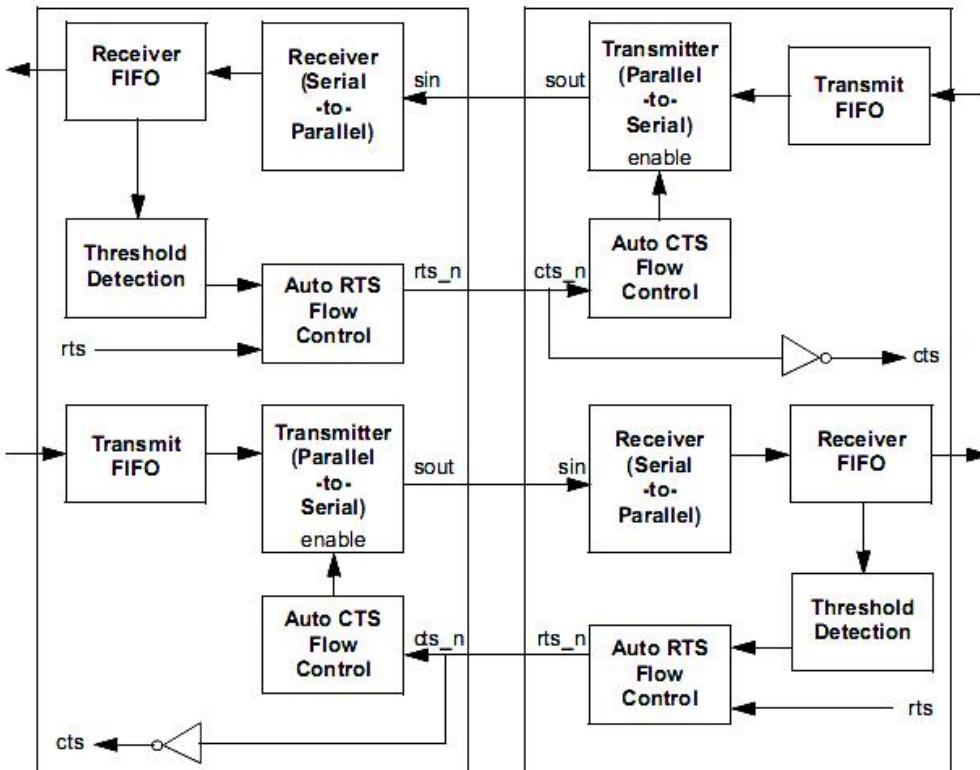


Fig. 53-5 UART Auto flow control block diagram

Auto RTS – Becomes active when the following occurs:

- Auto Flow Control is selected during configuration
- FIFOs are implemented
- RTS (MCR[1] bit and MCR[5]bit are both set)
- FIFOs are enabled (FCR[0] bit is set)
- SIR mode is disabled (MCR[6] bit is not set)

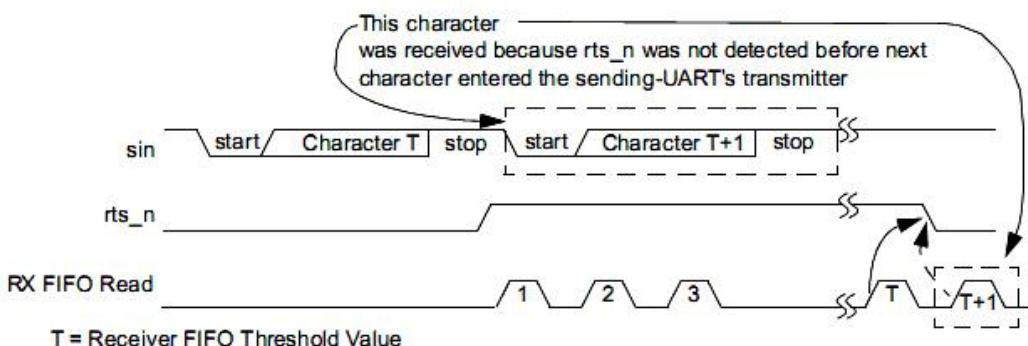


Fig. 53-6 UART AUTO RTS TIMING

Auto CTS – becomes active when the following occurs:

- Auto Flow Control is selected during configuration
- FIFOs are implemented
- AFCE (MCR[5] bit is set)
- FIFOs are enabled through FIFO Control Register FCR[0] bit
- SIR mode is disabled (MCR[6] bit is not set)

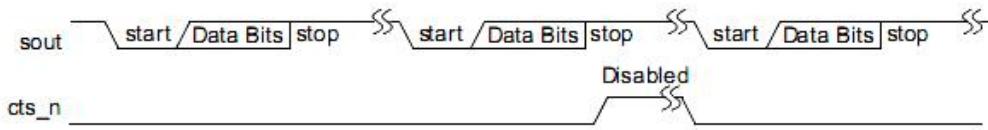


Fig. 53-7 UART AUTO CTS TIMING

## 53.4 Register Description

This section describes the control/status registers of the design. There are 5 UARTs in RK3288, and each one has its own base address.

### 53.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
UART_RBR	0x0000	W	0x00000000	Receive Buffer Register
UART_THR	0x0000	W	0x00000000	Transmit Holding Register
UART_DLL	0x0000	W	0x00000000	Divisor Latch (Low)
UART_DLH	0x0004	W	0x00000000	Divisor Latch (High)
UART_IER	0x0004	W	0x00000000	Interrupt Enable Register
UART_IIR	0x0008	W	0x00000001	Interrupt Identification Register
UART_FCR	0x0008	W	0x00000000	FIFO Control Register
UART_LCR	0x000c	W	0x00000000	Line Control Register
UART_MCR	0x0010	W	0x00000000	Modem Control Register
UART_LSR	0x0014	W	0x00000060	Line Status Register
UART_MSR	0x0018	W	0x00000000	Modem Status Register
UART_SCR	0x001c	W	0x00000000	Scratchpad Register
UART_SRBR	0x0030~0x006c	W	0x00000000	Shadow Receive Buffer Register
UART_STHR	0x0030~0x006c	W	0x00000000	Shadow Transmit Holding Register
UART_FAR	0x0070	W	0x00000000	FIFO Access Register
UART_TFR	0x0074	W	0x00000000	Transmit FIFO Read
UART_RFW	0x0078	W	0x00000000	Receive FIFO Write
UART_USR	0x007c	W	0x00000006	UART Status Register
UART_TFL	0x0080	W	0x00000000	Transmit FIFO Level
UART_RFL	0x0084	W	0x00000000	Receive FIFO Level
UART_SRR	0x0088	W	0x00000000	Software Reset Register
UART_SRTS	0x008c	W	0x00000000	Shadow Request to Send
UART_SBCR	0x0090	W	0x00000000	Shadow Break Control Register
UART_SDMAM	0x0094	W	0x00000000	Shadow DMA Mode

Name	Offset	Size	Reset Value	Description
UART_SFE	0x0098	W	0x00000000	Shadow FIFO Enable
UART_SRT	0x009c	W	0x00000000	Shadow RCVR Trigger
UART_STET	0x00a0	W	0x00000000	Shadow TX Empty Trigger
UARTHTX	0x00a4	W	0x00000000	Halt TX
UART_DMASA	0x00a8	W	0x00000000	DMA Software Acknowledge
UART_CPR	0x00f4	W	0x00000000	Component Parameter Register
UART_UCV	0x00f8	W	0x3330382a	UART Component Version
UART_CTR	0x00fc	W	0x44570110	Component Type Register

Notes: Size: **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** -WORD (32 bits) access

### 53.4.2 Detail Register Description

#### UART\_RBR

Address: Operational Base + offset (0x0000)

Receive Buffer Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	<p><b>data_input</b>            Data byte received on the serial input port (sin) in UART mode, or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line Status Register (LCR) is set.</p> <p>If in non-FIFO mode (FIFO_MODE == NONE) or FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an over-run error.</p> <p>If in FIFO mode (FIFO_MODE != NONE) and FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO is preserved, but any incoming data are lost and an over-run error occurs.</p>

#### UART\_THR

Address: Operational Base + offset (0x0000)

Transmit Holding Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	RW	0x00	<p>data_output</p> <p>Data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set.</p> <p>If in non-FIFO mode or FIFOs are disabled (FCR[0] = 0) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten.</p> <p>If in FIFO mode and FIFOs are enabled (FCR[0] = 1) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p>

**UART\_DLL**

Address: Operational Base + offset (0x0000)

Divisor Latch (Low)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x00	<p>baud_rate_divisor_L</p> <p>Lower 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. This register may only be accessed when the DLAB bit (LCR[7]) is set and the UART is not busy (USR[0] is zero). The output baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock frequency) / (16 * divisor).</p> <p>Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications occur. Also, once the DLH is set, at least 8 clock cycles of the slowest UART clock should be allowed to pass before transmitting or receiving data.</p>

**UART\_DLH**

Address: Operational Base + offset (0x0004)

Divisor Latch (High)

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x00	baud_rate_divisor_H Upper 8 bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.

**UART\_IER**

Address: Operational Base + offset (0x0004)

Interrupt Enable Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7	RW	0x0	prog_thre_int_en Programmable THRE Interrupt Mode Enable This is used to enable/disable the generation of THRE Interrupt. 1'b0: disabled 1'b1: enabled
6:4	RO	0x0	reserved
3	RW	0x0	modem_status_int_en Enable Modem Status Interrupt. This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt. 1'b0: disabled 1'b1: enabled
2	RW	0x0	receive_line_status_int_en Enable Receiver Line Status Interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt. 1'b0: disabled 1'b1: enabled
1	RW	0x0	trans_hold_empty_int_en Enable Transmit Holding Register Empty Interrupt.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
0	RW	0x0	<p>receive_data_available_int_en Enable Received Data Available Interrupt. This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt (if in FIFO mode and FIFOs enabled). These are the second highest priority interrupts.</p> <p>1'b0: disabled 1'b1: enabled</p>

**UART\_IIR**

Address: Operational Base + offset (0x0008)

Interrupt Identification Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:6	RO	0x0	<p>fifos_en FIFOs Enabled. This is used to indicate whether the FIFOs are enabled or disabled.</p> <p>2'b00: disabled 2'b11: enabled</p>
5:4	RO	0x0	reserved
3:0	RO	0x1	<p>int_id Interrupt ID This indicates the highest priority pending interrupt which can be one of the following types:</p> <p>4'b0000: modem status 4'b0001: no interrupt pending 4'b0010: THR empty 4'b0100: received data available 4'b0110: receiver line status 4'b0111: busy detect 4'b1100: character timeout</p>

**UART\_FCR**

Address: Operational Base + offset (0x0008)

FIFO Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:6	WO	0x0	<p>rcvr_trigger RCVR Trigger.</p> <p>This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated. In auto flow control mode it is used to determine when the rts_n signal is de-asserted. It also determines when the dma_rx_req_n signal is asserted in certain modes of operation. The following trigger levels are supported:</p> <ul style="list-style-type: none"> <li>2'b00: 1 character in the FIFO</li> <li>2'b01: FIFO 1/4 full</li> <li>2'b10: FIFO 1/2 full</li> <li>2'b11: FIFO 2 less than full</li> </ul>
5:4	WO	0x0	<p>tx_empty_trigger TX Empty Trigger.</p> <p>This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active. It also determines when the dma_tx_req_n signal is asserted when in certain modes of operation.</p> <p>The following trigger levels are supported:</p> <ul style="list-style-type: none"> <li>2'b00: FIFO empty</li> <li>2'b01: 2 characters in the FIFO</li> <li>2'b10: FIFO 1/4 full</li> <li>2'b11: FIFO 1/2 full</li> </ul>
3	WO	0x0	<p>dma_mode DMA Mode</p> <p>This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals when additional DMA handshaking signals are not selected .</p> <ul style="list-style-type: none"> <li>1'b0: mode 0</li> <li>1'b1: mode 11100 = character timeout.</li> </ul>
2	WO	0x0	<p>xmit_fifo_reset XMIT FIFO Reset.</p> <p>This resets the control portion of the transmit FIFO and treats the FIFO as empty. This also de-asserts the DMA TX request and single signals when additional DMA handshaking signals are select. Note that this bit is 'self-clearing'. It is not necessary to clear this bit.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	WO	0x0	rcvr_fifo_reset RCVR FIFO Reset. This resets the control portion of the receive FIFO and treats the FIFO as empty. This also de-asserts the DMA RX request and single signals when additional DMA handshaking signals are selected. Note that this bit is 'self-clearing'. It is not necessary to clear this bit.
0	WO	0x0	fifo_en FIFO Enable. FIFO Enable. This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs. Whenever the value of this bit is changed both the XMIT and RCVR controller portion of FIFOs is reset.

**UART\_LCR**

Address: Operational Base + offset (0x000c)

Line Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7	RW	0x0	div_lat_access Divisor Latch Access Bit. Writable only when UART is not busy (USR[0] is zero), always readable. This bit is used to enable reading and writing of the Divisor Latch register (DLL and DLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers.
6	RW	0x0	break_ctrl Break Control Bit. This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared. If MCR[6] set to one, the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low.
5	RO	0x0	reserved

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
4	RW	0x0	<p>even_parity_sel Even Parity Select. Writeable only when UART is not busy (USR[0] is zero), always readable. This is used to select between even and odd parity, when parity is enabled (PEN set to one). If set to one, an even number of logic 1s is transmitted or checked. If set to zero, an odd number of logic 1s is transmitted or checked.</p>
3	RW	0x0	<p>parity_en Parity Enable. Writeable only when UART is not busy (USR[0] is zero), always readable. This bit is used to enable and disable parity generation and detection in transmitted and received serial character respectively. 1'b0: parity disabled 1'b1: parity enabled</p>
2	RW	0x0	<p>stop_bits_num Number of stop bits. Writeable only when UART is not busy (USR[0] is zero), always readable. This is used to select the number of stop bits per character that the peripheral transmits and receives. If set to zero, one stop bit is transmitted in the serial data. If set to one and the data bits are set to 5 (LCR[1:0] set to zero) one and a half stop bits is transmitted. Otherwise, two stop bits are transmitted. Note that regardless of the number of stop bits selected, the receiver checks only the first stop bit. 1'b0: 1 stop bit 1'b1: 1.5 stop bits when DLS (LCR[1:0]) is zero, else 2 stop bit.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1:0	RW	0x0	<p>data_length_sel Data Length Select. Writable only when UART is not busy (USR[0] is zero), always readable. This is used to select the number of data bits per character that the peripheral transmits and receives. The number of bit that may be selected areas follows:</p> <ul style="list-style-type: none"> <li>2'b00: 5 bits</li> <li>1'b01: 6 bits</li> <li>1'b10: 7 bits</li> <li>1'b11: 8 bits</li> </ul>

**UART\_MCR**

Address: Operational Base + offset (0x0010)

Modem Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:7	RO	0x0	reserved
6	RW	0x0	<p>sir_mode_en SIR Mode Enable. SIR Mode Enable. This is used to enable/disable the IrDA SIR Mode.</p> <ul style="list-style-type: none"> <li>1'b0: IrDA SIR Mode disabled</li> <li>1'b1: IrDA SIR Mode enabled</li> </ul>
5	RW	0x0	<p>auto_flow_ctrl_en Auto Flow Control Enable. 1'b0: Auto Flow Control Mode disabled 1'b1: Auto Flow Control Mode enabled</p>
4	RW	0x0	<p>loopback LoopBack Bit. This is used to put the UART into a diagnostic mode for test purposes.</p>
3	RW	0x0	<p>out2 OUT2. This is used to directly control the user-designated Output2 (out2_n) output. The value written to this location is inverted and driven out on out2_n, that is:</p> <ul style="list-style-type: none"> <li>1'b0: out2_n de-asserted (logic 1)</li> <li>1'b1: out2_n asserted (logic 0)</li> </ul>
2	RW	0x0	<p>out1 OUT1</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RW	0x0	<p>req_to_send Request to Send.</p> <p>This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART is ready to exchange data.</p>
0	RW	0x0	<p>data_terminal_ready Data Terminal Ready.</p> <p>This is used to directly control the Data Terminal Ready (dtr_n) output. The value written to this location is inverted and driven out on dtr_n, that is:</p> <p>1'b0: dtr_n de-asserted (logic 1) 1'b1: dtr_n asserted (logic 0)</p>

**UART\_LSR**

Address: Operational Base + offset (0x0014)

Line Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7	RO	0x0	<p>receiver_fifo_error Receiver FIFO Error bit.</p> <p>This bit is relevant FIFOs are enabled (FCR[0] set to one). This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO.</p> <p>1'b0: no error in RX FIFO 1'b1: error in RX FIFO</p>
6	RO	0x1	<p>trans_empty Transmitter Empty bit.</p> <p>Transmitter Empty bit. If FIFOs enabled (FCR[0] set to one), this bit is set whenever the Transmitter Shift Register and the FIFO are both empty. If FIFOs are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty.</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
5	RO	0x1	<p>trans_hold_reg_empty Transmit Holding Register Empty bit. If THRE mode is disabled (IER[7] set to zero) and regardless of FIFO's being implemented/enabled or not, this bit indicates that the THR or TX FIFO is empty.</p> <p>This bit is set whenever data is transferred from the THR or TX FIFO to the transmitter shift register and no new data has been written to the THR or TX FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. If IER[7] set to one and FCR[0] set to one respectively, the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] threshold setting.</p>
4	RO	0x0	<p>break_int Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data.</p>
3	RO	0x0	<p>framing_error Framing Error bit. This is used to indicate the occurrence of a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data.</p>
2	RO	0x0	<p>parity_error Parity Error bit. This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (LCR[3]) is set.</p>
1	RO	0x0	<p>overrun_error Overrun error bit. This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read.</p>
0	RO	0x0	<p>data_ready Data Ready bit. This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO. 1'b0: no data ready 1'b1: data ready</p>

**UART\_MSR**

Address: Operational Base + offset (0x0018)

Modem Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7	RO	0x0	data_carrier_detect Data Carrier Detect. This is used to indicate the current state of the modem control line dcd_n.
6	RO	0x0	ring_indicator Ring Indicator. This is used to indicate the current state of the modem control line ri_n.
5	RO	0x0	data_set_ready Data Set Ready. This is used to indicate the current state of the modem control line dsr_n.
4	RO	0x0	clear_to_send Clear to Send. This is used to indicate the current state of the modem control line cts_n.
3	RO	0x0	delta_data_carrier_detect Delta Data Carrier Detect. This is used to indicate that the modem control line dcd_n has changed since the last time the MSR was read.
2	RO	0x0	trailing_edge_ring_indicator Trailing Edge of Ring Indicator. Trailing Edge of Ring Indicator. This is used to indicate that a change on the input ri_n (from an active-low to an inactive-high state) has occurred since the last time the MSR was read.
1	RO	0x0	delta_data_set_ready Delta Data Set Ready. This is used to indicate that the modem control line dsr_n has changed since the last time the MSR was read.
0	RO	0x0	delta_clear_to_send Delta Clear to Send. This is used to indicate that the modem control line cts_n has changed since the last time the MSR was read.

**UART\_SCR**

Address: Operational Base + offset (0x001c)

## Scratchpad Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RW	0x00	This register is for programmers to use as a temporary storage space.

**UART\_SRBR**

Address: Operational Base + offset (0x0030~0x006c)

## Shadow Receive Buffer Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RO	0x00	<p>shadow_rbr</p> <p>This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set.</p> <p>If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an overrun error.</p> <p>If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO are preserved, but any incoming data is lost. An overrun error also occurs.</p>

**UART\_STHR**

Address: Operational Base + offset (0x0030~0x006c)

## Shadow Transmit Holding Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RO	0x00	<p>shadow_thr</p> <p>This is a shadow register for the THR.</p>

**UART\_FAR**

Address: Operational Base + offset (0x0070)

## FIFO Access Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	<p>fifo_access_test_en</p> <p>This register is use to enable a FIFO access mode for testing, so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master when FIFOs are implemented and enabled. When FIFOs are not enabled it allows the RBR to be written by the master and the THR to be read by the master.</p> <p>1'b0: FIFO access mode disabled 1'b1: FIFO access mode enabled</p>

**UART\_TFR**

Address: Operational Base + offset (0x0074)

Transmit FIFO Read

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:8	RO	0x0	reserved
7:0	RO	0x00	<p>trans_fifo_read</p> <p>Transmit FIFO Read.</p> <p>These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one).When FIFOs are implemented and enabled, reading this register gives the data at the top of the transmit FIFO. Each consecutive read pops the transmit FIFO and gives the next data value that is currently at the top of the FIFO.</p>

**UART\_RFW**

Address: Operational Base + offset (0x0078)

Receive FIFO Write

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:10	RO	0x0	reserved
9	WO	0x0	<p>receive_fifo_framing_error</p> <p>Receive FIFO Framing Error.</p> <p>These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one).</p>
8	WO	0x0	<p>receive_fifo_parity_error</p> <p>Receive FIFO Parity Error.</p> <p>These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one).</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
7:0	WO	0x00	<p>receive_fifo_write Receive FIFO Write Data.</p> <p>These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFOs are enabled, the data that is written to the RFWD is pushed into the receive FIFO. Each consecutive write pushes the new data to the next write location in the receive FIFO. When FIFOs not enabled, the data that is written to the RFWD is pushed into the RBR.</p>

**UART\_USR**

Address: Operational Base + offset (0x007c)

UART Status Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4	RO	0x0	<p>receive_fifo_full Receive FIFO Full.</p> <p>This is used to indicate that the receive FIFO is completely full.</p> <p>1'b0: Receive FIFO not full 1'b1: Receive FIFO Full</p> <p>This bit is cleared when the RX FIFO is no longer full.</p>
3	RO	0x0	<p>receive_fifo_not_empty Receive FIFO Not Empty.</p> <p>This is used to indicate that the receive FIFO contains one or more entries.</p> <p>1'b0: Receive FIFO is empty 1'b1: Receive FIFO is not empty</p> <p>This bit is cleared when the RX FIFO is empty.</p>
2	RO	0x1	<p>trans_fifo_empty Transmit FIFO Empty.</p> <p>This is used to indicate that the transmit FIFO is completely empty.</p> <p>1'b0: Transmit FIFO is not empty 1'b1: Transmit FIFO is empty</p> <p>This bit is cleared when the TX FIFO is no longer empty</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	RO	0x1	<p>trans_fifo_not_full Transmit FIFO Not Full. This is used to indicate that the transmit FIFO is not full.</p> <p>1'b0: Transmit FIFO is full 1'b1: Transmit FIFO is not full This bit is cleared when the TX FIFO is full.</p>
0	RO	0x0	<p>uart_busy UART Busy. UART Busy. This indicates that a serial transfer is in progress, when cleared indicates that the UART is idle or inactive.</p> <p>1'b0: UART is idle or inactive 1'b1: UART is busy (actively transferring data)</p>

**UART\_TFL**

Address: Operational Base + offset (0x0080)

Transmit FIFO Level

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RW	0x00	<p>trans_fifo_level Transmit FIFO Level. This indicates the number of data entries in the transmit FIFO.</p>

**UART\_RFL**

Address: Operational Base + offset (0x0084)

Receive FIFO Level

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:5	RO	0x0	reserved
4:0	RO	0x00	<p>receive_fifo_level Receive FIFO Level. This indicates the number of data entries in the receive FIFO.</p>

**UART\_SRR**

Address: Operational Base + offset (0x0088)

Software Reset Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2	WO	0x0	<p>xmit_fifo_reset XMIT FIFO Reset. This is a shadow register for the XMIT FIFO Reset bit (FCR[2]).</p>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
1	WO	0x0	rcvr_fifo_reset RCVR FIFO Reset. This is a shadow register for the RCVR FIFO Reset bit (FCR[1]).
0	WO	0x0	uart_reset UART Reset. This asynchronously resets the UART and synchronously removes the reset assertion. For a two clock implementation both pclk and sclk domains are reset.

**UART\_SRTS**

Address: Operational Base + offset (0x008c)

Shadow Request to Send

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	shadow_req_to_send Shadow Request to Send. This is a shadow register for the RTS bit (MCR[1]), this can be used to remove the burden of having to performing a read-modify-write on the MCR.

**UART\_SBCR**

Address: Operational Base + offset (0x0090)

Shadow Break Control Register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	shadow_break_ctrl Shadow Break Control Bit. This is a shadow register for the Break bit (LCR[6]), this can be used to remove the burden of having to performing a read modify write on the LCR.

**UART\_SDMAM**

Address: Operational Base + offset (0x0094)

Shadow DMA Mode

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	shadow_dma_mode Shadow DMA Mode. This is a shadow register for the DMA mode bit (FCR[3]).

**UART\_SFE**

Address: Operational Base + offset (0x0098)

## Shadow FIFO Enable

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	shadow_fifo_en Shadow FIFO Enable. Shadow FIFO Enable. This is a shadow register for the FIFO enable bit (FCR[0]).

**UART\_SRT**

Address: Operational Base + offset (0x009c)

Shadow RCVR Trigger

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
1:0	RW	0x0	shadow_rcvr_trigger Shadow RCVR Trigger. This is a shadow register for the RCVR trigger bits (FCR[7:6]).

**UART\_STET**

Address: Operational Base + offset (0x00a0)

Shadow TX Empty Trigger

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
1:0	RW	0x0	shadow_tx_empty_trigger Shadow TX Empty Trigger. This is a shadow register for the TX empty trigger bits (FCR[5:4]).

**UART\_HTX**

Address: Operational Base + offset (0x00a4)

Halt TX

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	halt_tx_en This register is use to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFOs are implemented and enabled. 1'b0: Halt TX disabled 1'b1: Halt TX enabled

**UART\_DMASA**

Address: Operational Base + offset (0x00a8)

DMA Software Acknowledge

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved

Bit	Attr	Reset Value	Description
0	WO	0x0	dma_software_ack This register is use to perform a DMA software acknowledge if a transfer needs to be terminated due to an error condition.

**UART\_UCV**

Address: Operational Base + offset (0x00f8)

UART Component Version

Bit	Attr	Reset Value	Description
31:0	RO	0x3330382a	ver ASCII value for each number in the version

**UART\_CTR**

Address: Operational Base + offset (0x00fc)

Component Type Register

Bit	Attr	Reset Value	Description
31:0	RO	0x44570110	peripheral_id This register contains the peripherals identification code.

**53.5 Interface description**

Table 53-1 UART Interface Description

Module pin	Direction	Pad name	IOMUX
UART_BT Interface			
uartbt_sin	I	GPIO4_C[0]	GPIO1A_IOMUX[0]=1
uartbt_sout	O	GPIO4_C[1]	GPIO1A_IOMUX[2]=1
uartbt_cts_n	I	GPIO4_C[2]	GPIO1A_IOMUX[4]=1
uartbt_rts_n	O	GPIO4_C[3]	GPIO1A_IOMUX[6]=1
UART_BB Interface			
uartbb_sin	I	GPIO5_B[0]	GPIO1A_IOMUX[1:0]=01
uartbb_sout	O	GPIO5_B[1]	GPIO1A_IOMUX[3:2]=01
uartbb_cts_n	I	GPIO5_B[2]	GPIO1A_IOMUX[5:4]=01
uartbb_rts_n	O	GPIO5_B[3]	GPIO1A_IOMUX[7:6]=01
UART_DBG Interface			
uartdbg_sin	I	GPIO7_C[6]	GPIO7CH_IOMUX[9:8]=01
uartdbg_sout	O	GPIO7_C[7]	GPIO7CH_IOMUX[14:12]=001
uartdbg_sirsin	I	GPIO7_C[6]	GPIO7CH_IOMUX[9:8]=10
Uartdbg_sirout	O	GPIO7_C[7]	GPIO7CH_IOMUX[14:12]=010
UART_GPS Interface			
uartgps_sin	I	GPIO7_A[7]	GPIO7A_IOMUX[15:14]=01
uartgps_sout	O	GPIO7_B[0]	GPIO7B_IOMUX[1:0]=01
uartgps_cts_n	I	GPIO7_B[1]	GPIO7B_IOMUX[3:2]=01
uartgps_rts_n	O	GPIO7_B[2]	GPIO7B_IOMUX[5:4]=01
UART_EXP Interface			
uartexp_sin	I	GPIO5_B[7]	GPIO5B_IOMUX[15:14]=11
uartexp_sout	O	GPIO5_B[6]	GPIO5B_IOMUX[13:12]=11
uartexp_cts_n	I	GPIO5_B[4]	GPIO5B_IOMUX[9:8]=11
uartexp_rts_n	O	GPIO5_B[5]	GPIO5B_IOMUX[11:10]=11

**53.6 Ap**

## Application Notes

### 53.6.1 None FIFO Mode Transfer Flow

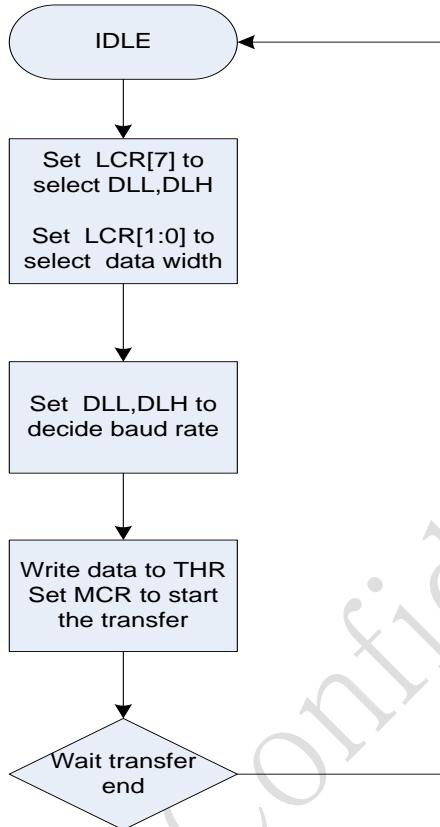


Fig. 53-8 UART none fifo mode

### 53.6.2 FIFO Mode Transfer Flow

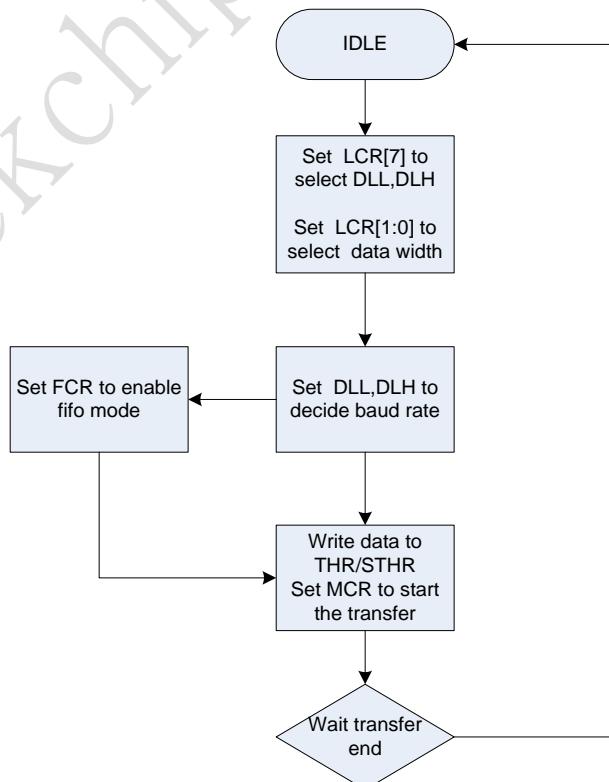


Fig. 53-9 UART fifo mode

The UART is an APB slave performing:

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 APB interface. The transmitting and receiving paths are buffered with internal FIFO memories enabling up to 64-bytes to be stored independently in both transmit and receive modes. A baud rate generator can generate a common transmit and receive internal clock input. The baud rates will depend on the internal clock frequency. The UART will also provide transmit, receive and exception interrupts to system. A DMA interface is implemented for improving the system performance.

### 53.6.3 Baud Rate Calculation

#### UART clock generation

The following figures shows the UART clock generation.

UART source clocks can be selected from CODEC PLL and GENERAL PLL outputs. UART\_BT source clocks can also be selected from NEW PLL and USBPHY 480M. UART clocks can be generated by 1 to 64 division of its source clock, or can be fractionally divided again, or be provided by XIN24M.

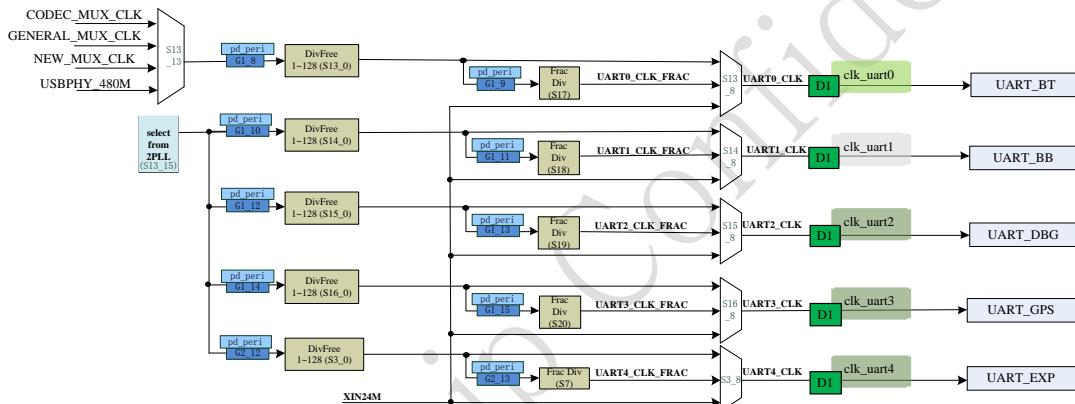


Fig. 53-10 UART clock generation

#### UART baud rate configuration

The following table provides some reference configuration for different UART baud rates.

Table 53-2 UART baud rate configuration

Baud Rate	Reference Configuration
115.2 Kbps	Configure GENERAL PLL to get 648MHz clock output; Divide 648MHz clock by 1152/50625 to get 14.7456MHz clock; Config UART_DLL to 8.
460.8 Kbps	Configure GENERAL PLL to get 648MHz clock output; Divide 648MHz clock by 1152/50625 to get 14.7456MHz clock; Configure UART_DLL to 2.
921.6 Kbps	Configure GENERAL PLL to get 648MHz clock output; Divide 648MHz clock by 1152/50625 to get 14.7456MHz clock; Configure UART_DLL to 1.
1.5 Mbps	Choose GENERAL PLL to get 384MHz clock output; Divide 384MHz clock by 16 to get 24MHz clock; Configure UART_DLL to 1
3 Mbps	Choose GENERAL PLL to get 384MHz clock output; Divide 384MHz clock by 8 to get 48MHz clock; Configure UART_DLL to 1
4 Mbps	Configure GENERAL PLL to get 384MHz clock output;

	Divide 384MHz clock by 6 to get 64MHz clock; Configure UART_DLL to 1
--	---

### 53.6.4 CTS\_n and RTS\_n Polarity Configurable

The polarity of cts\_n and rts\_n ports can be configured by GRF registers.

- GRF\_SOC\_CON13[4:0] (grf\_uart\_cts\_sel[4:0]) used to configure the polarity of cts\_n. Every bit for one UART, bit4 is for UART\_EXP, bit3 is for UART\_GPS, bit2 is for UART\_DBG, bit1 is for UART\_BB, bit0 is for UART\_BT.
- GRF\_SOC\_CON13[9:5] (grf\_uart\_rts\_sel[4:0]) used to configure the polarity of rts\_n. Every bit for one UART, bit4 is for UART\_EXP, bit3 is for UART\_GPS, bit2 is for UART\_DBG, bit1 is for UART\_BB, bit0 is for UART\_BT.
- When grf\_uart\_cts\_sel[\*] is configured as 1'b1, cts\_n is high active. Otherwise, low active.
- When grf\_uart\_rts\_sel[\*] is configured as 1'b1, rts\_n is high active. Otherwise, low active.

## Chapter 54 Process-Voltage-Temperature Monitor (PVTM)

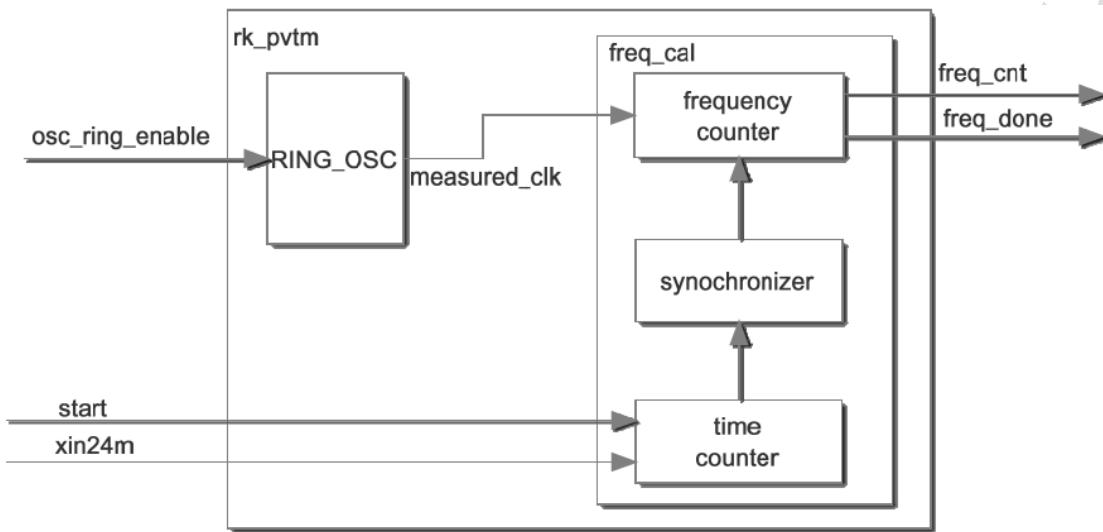
### 54.1 Overview

The Process-Voltage-Temperature Monitor (PVTM) is used to monitor the chip performance variance caused by chip process, voltage and temperature.

PVTM supports the following features:

- a clock oscillation ring is integrated and used to generate a clock like signal, the frequency of this clock is determined by the cell delay value of clock oscillation ring circuit
- a frequency counter is used to measure the frequency of the clock oscillation ring.

### 54.2 Block Diagram



The PVTM include two main blocks

- RING\_OSC, it is composed with inverters with odd number, which is used to generate a clock
- Freq\_cal, it is used to measure the frequency of clock which generated from the RING\_SOC block

### Frequency Calculation

A frequency fixed clock(24MH) is used to calculate the clock cycles of RING\_OSC generated clock. Suppose the time period is 1s, then the clock period of RING\_OSC clock is  $T = 1/2 * \text{clock\_counter}(s)$

## Chapter 55 Memory-Management-Unit (MMU)

### 55.1 Overview

An MMU controls address translation, access permissions, memory attribute.determination, and checking at a memory system level.

There are two types of MMU in RK3288.

The MMU used in pd\_peri can be referred to the document DDI0472A\_corelink\_mmu\_400\_r0p0\_trm.pdf.

The MMU used in other modules will be introduced below.

### 55.2 Block Diagram

The MMU divides memory into 4KB pages, where each page can be individually configured.

The MMU uses a 2-level page table structure:

1. The first level, the Page Directory consists of 1024 Directory Table Entries (DTEs), each pointing to a Page Table.
2. The second level, the Page Table consists of 1024 Page Table Entries (PTEs), each pointing to a page in memory

Fig. 56-1 shows the structure of the two-level page table.

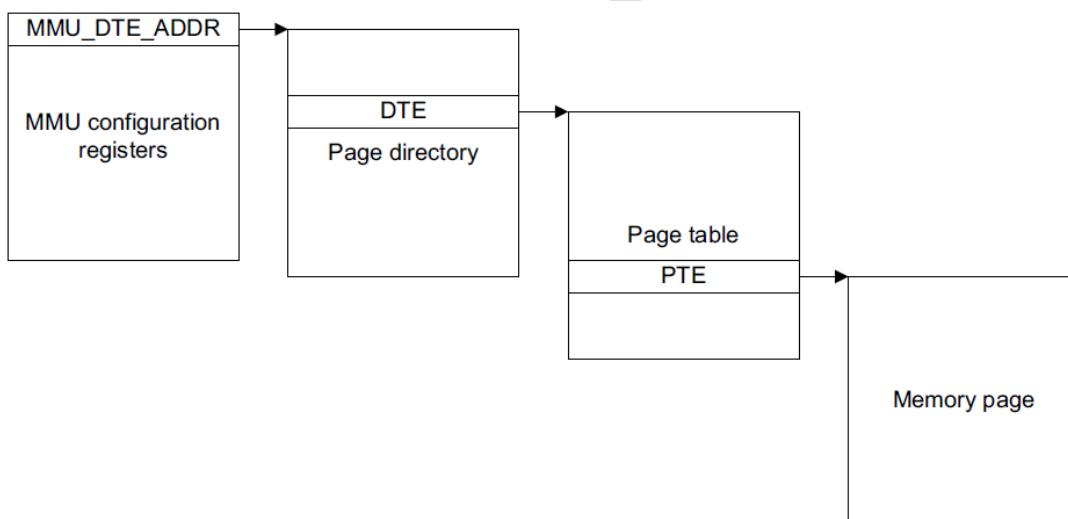


Fig. 53-11 Power Domain Partition

Fig. 56-2 shows the arrangement of the MMU address bits.

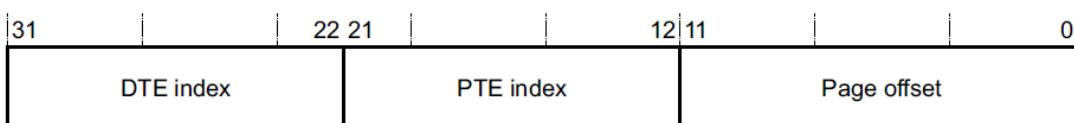


Fig. 56-2 Power Domain Partition

The MMU uses the following algorithm to translate an address:

- Find the DTE at address given by:

`MMU_DTE_ADDR + (4 * DTE index)`

- Find the PTE at address given by:

Page table address from DTE + (4 \* PTE index)

- Calculate effective address as follows:

Page address from PTE + Page offset.

The page directory is a 4KB data structure that contains 1024 32-bit DTEs. The page directory must align at a 4KB boundary in memory.

Each DTE contains the address of a page table and a page table present bit. The system:

- initializes the entire page directory before use
- clears the page table present bit for any DTE that does not point to a valid page table.

Fig. 56-3 shows the page bit assignments.

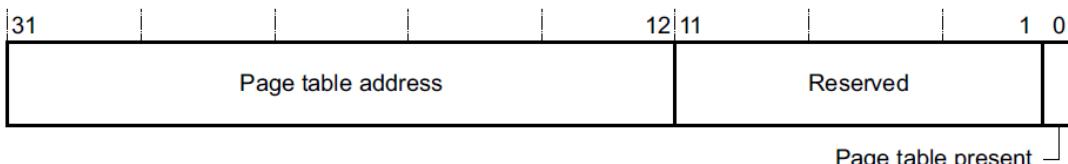


Fig. 56-3 Page directory entry bit assignments

Bits	Name	Function
[31:12]	Page table address	This field stores bits [31:12] of the address for a page table
[11:1]	Reserved	Reserved, write as zero
[0]	Page table present	This bit indicates when the page table address points to a valid page table. 0 = page table not valid 1 = page table valid.

The page table is a 4KB data structure containing 1024 32-bit PTEs. The page table must be aligned at a 4KB boundary in memory.

Each PTE contains the address of a page of memory, a Page Table present bit, and Read/Write Permission bits. The entire Page Table must be initialized before use, and any PTE not pointing to a valid page must clear the Page Present bit.

Fig.56-3 on page 3-233 shows the page table entry bit assignments.

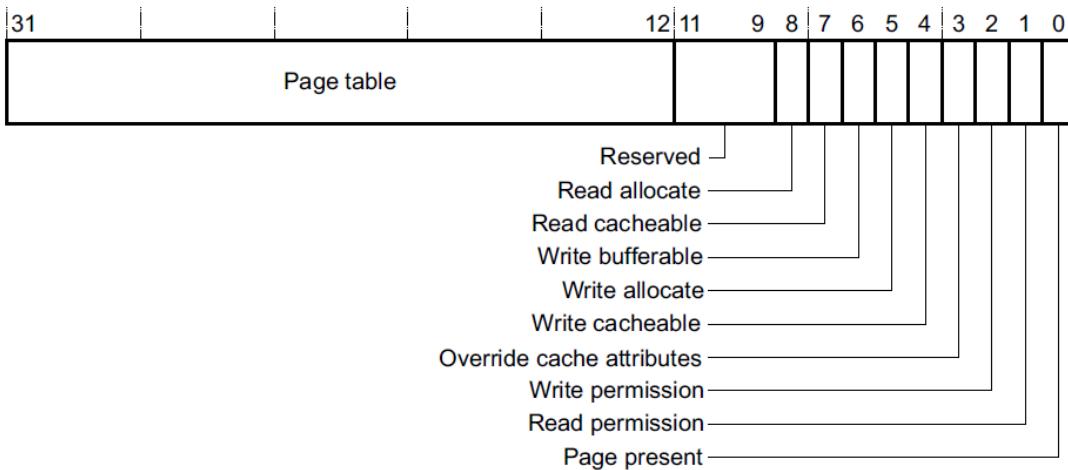


Fig. 56-3 Page directory entry bit assignments

Bits	Name	Function
[31:12]	Page table address	This field stores bits [31:12] of the address for a page table
[11:9]	Reserved	Reserved, write as zero
[8]	Read allocate	If set, allocate cache space on read misses. Must not be set if the Read cacheable bit is not set. Only used for reads, if the Override cache attributes bit is set.
[7]	Read cacheable	If set, enable caching or prefetching of data. Only used for reads, if the Override cache attributes bit is set.
[6]	Write bufferable	If set, enable write to be delayed on their way to memory. Only used for writes, if the Override cache attributes bit is set.
[5]	Write allocate	If set, allocate cache space on write misses. Must not be set if the Write cacheable bit is not set. Only used for writes, if the Override cache attributes is set.
[4]	Write cacheable	If set, enable different writes to be merged together. Only used for writes, if the Override cache attributes bit is set.
[3]	Override cache attributes	If set, the cacheability attributes specified in bits [8:4] are used to control the cache attributes used on the memory bus. If cleared, the default cacheability attributes from the specific processors are used on the system bus.
[2]	Write permission	Enable write accesses to the page, if present.
[1]	Read permission	Enable read accesses from the page, if present.
[0]	Page present	This bit indicates when the page table field points to a valid page. 0 = page not valid 1 = page valid.

## 55.3 Register Description

### 55.3.1 Register Summary

Name	Offset	Size	Reset Value	Description
MMU_DTE_ADDR	0x0000	W	0x00000000	MMU current page table address
MMU_STATUS	0x0004	W	0x00000018	MMU status register
MMU_CMD	0x0008	W	0x00000000	MMU command register
MMU_PAGE_FAULT_ADDR	0x000c	W	0x00000000	MMU logic address of last page fault register
MMU_ZAP_ONE_LINE	0x0010	W	0x00000000	MMU zap cache line register
MMU_INT_RAWSTAT	0x0014	W	0x00000000	MMU raw interrupt status register
MMU_INT_CLEAR	0x0018	W	0x00000000	MMU interrupt clear register
MMU_INT_MASK	0x001c	W	0x00000000	MMU interrupt mask register
MMU_INT_STATUS	0x0020	W	0x00000000	MMU interrupt status register
MMU_AUTO_GATING	0x0024	W	0x00000000	clock atuo gating register

Notes: Size : **B** - Byte (8 bits) access, **HW** - Half WORD (16 bits) access, **W** - WORD (32 bits) access

### 55.3.2 Detail Register Description

#### MMU\_DTE\_ADDR

Address: Operational Base + offset (0x0000)

MMU current page table address

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	mmu_dte_addr page table address

#### MMU\_STATUS

Address: Operational Base + offset (0x0004)

MMU status register

Bit	Attr	Reset Value	Description
31:11	RO	0x0	reserved
10:6	RO	0x00	mmu_page_fault_bus_id Index of master responsible for the last page fault
5	RO	0x0	mmu_page_fault_is_write The direction of access for last page fault: 0: read 1:write
4	RO	0x1	mmu_replay_buffer_empty The MMU replay buffer is empty.

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
3	RO	0x1	mmu_idle the MMU is idle when accesses are being translated and there are no unfinished translated access. The MMU_IDLE signal only reports idle when the MMU processor is idle and accesses are active on the external bus. Note: the MMU can be idle in page fault mode.
2	RO	0x0	mmu_stall_active MMU stall mode currently enabled. The mode is enabled by command.
1	RO	0x0	mmu_page_fault_active MMU page fault mode currently enabled. The mode is enabled by command
0	RO	0x0	mmu_paging_enabled mmu paging is enabled.

**MMU\_CMD**

Address: Operational Base + offset (0x0008)

MMU command register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:3	RO	0x0	reserved
2:0	RW	0x0	mmu_cmd 0: MMU_ENABLE_PAGING. enable paging. 1: MMU_DISABLE_PAGING. disable paging. 2: MMU_ENABLE_STALL. turn on stall mode. 3: MMU_DISABLE_STALL. turn off stall mode. 4: MMU_ZAP_CACHE. zap the entire page table cache. 5: MMU_PAGE_FAULT_DONE. leave page fault mode. 6: MMU_FORCE_RESET. reset the mmu. The MMU_ENABLE_STALL command can always be issued. Other commands are ignored unless the MMU is idle or stalled.

**MMU\_PAGE\_FAULT\_ADDR**

Address: Operational Base + offset (0x000c)

MMU logic address of last page fault register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RO	0x00000000	mmu_page_fault_addr address of last page fault

**MMU\_ZAP\_ONE\_LINE**

Address: Operational Base + offset (0x0010)

MMU zap cache line register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:0	RW	0x00000000	mmu_zap_one_line address to be invalidated from the page table cache.

**MMU\_INT\_RAWSTAT**

Address: Operational Base + offset (0x0014)

MMU raw interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RO	0x0	read_bus_error read bus error
0	RO	0x0	page_fault page fault

**MMU\_INT\_CLEAR**

Address: Operational Base + offset (0x0018)

MMU interrupt clear register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	read_bus_error_clear read bus error interrupt clear. write 1 to this register can clear read bus error interrupt.
0	RW	0x0	page_fault_clear page fault interrupt clear, write 1 to this register can clear page fault interrupt.

**MMU\_INT\_MASK**

Address: Operational Base + offset (0x001c)

MMU interrupt mask register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	read_bus_error_int_en read bus error interrupt enable
0	RW	0x0	page_fault_int_en page fault interrupt enable

**MMU\_INT\_STATUS**

Address: Operational Base + offset (0x0020)

MMU interrupt status register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:2	RO	0x0	reserved
1	RW	0x0	read_bus_error read bus error interrupt
0	RO	0x0	page_fault page fault interrupt

**MMU\_AUTO\_GATING**

Address: Operational Base + offset (0x0024)

clock atuo gating register

<b>Bit</b>	<b>Attr</b>	<b>Reset Value</b>	<b>Description</b>
31:1	RO	0x0	reserved
0	RW	0x0	mmu_atuo_gating mmu clock auto gating when it is 1, the mmu will auto gating itself

## 55.4 MMU Base address

The table below shows the MMU base address in different modules.

ISP_MMU0_BASE	ISP_BASEADDR + 0x4000
ISP_MMU1_BASE	ISP_BASEADDR + 0x5000
VOP_BIG_MMU_BASE	VOP_BIG_BASEADDR + 0x300
VOP_LIT_MMU_BASE	VOP_LIT_BASEADDR + 0x300
IEP_MMU_BASE	IEP_BASE + 0x800
VIP_MMU_BASE	VIP_BASE + 0x800