

Rockchip
RK3228A/RK3228B
Technical Reference Manual

Revision 1.0
Jan.2016

Revision History

| Date | Revision | Description |
|-------------|-----------------|--------------------|
| 2016-1-29 | 1.0 | Update |
| 2015-10-16 | 0.1 | Initial Release |

Rockchip Confidential

Table of Content

| | |
|---|-----|
| Table of Content | 3 |
| Figure Index | 6 |
| Table Index..... | 9 |
| Warranty Disclaimer..... | 11 |
| Chapter 1 Mobile Storage Host Controller..... | 12 |
| 1.1 Overview | 12 |
| 1.2 Block Diagram | 12 |
| 1.3 Function Description | 13 |
| 1.4 Register Description..... | 32 |
| 1.5 Interface Description | 63 |
| 1.6 Application Notes | 64 |
| Chapter 2 USB OTG 2.0 | 87 |
| 2.1 Overview | 87 |
| 2.2 Block Diagram | 87 |
| 2.3 USB OTG2.0 Controller..... | 89 |
| 2.4 USB OTG2.0 PHY..... | 92 |
| 2.5 UART BYPASS FUNCITON..... | 95 |
| 2.6 Register Description..... | 97 |
| 2.7 Interface description..... | 205 |
| 2.8 Application Note..... | 205 |
| Chapter 3 USB Host | 206 |
| 3.1 Overview | 206 |
| 3.2 Block Diagram | 206 |
| 3.3 USB Host2.0 Controller | 206 |
| 3.4 USB Host2.0 PHY | 206 |
| 3.5 Register Description..... | 206 |
| 3.6 Interface description..... | 212 |
| 3.7 Application Note..... | 212 |
| Chapter 4 HDMI TX | 213 |
| 4.1 Overview | 213 |
| 4.2 Block Diagram | 213 |
| 4.3 Function Description | 213 |
| 4.4 HDMI PHY..... | 217 |
| 4.5 Register Description..... | 227 |
| 4.6 Application Notes | 228 |
| Chapter 5 HDCP2.2 Controller | 239 |
| 5.1 Overview | 239 |
| 5.2 Block Diagram | 239 |
| 5.3 Function Description | 239 |
| 5.4 Register Description..... | 242 |
| 5.5 Application Notes | 247 |
| Chapter 6 Pulse Width Modulation (PWM)..... | 252 |
| 6.1 Overview | 252 |
| 6.2 Block Diagram | 252 |
| 6.3 Function Description | 253 |
| 6.4 Register Description..... | 254 |
| 6.5 Interface Description | 268 |
| Application Notes..... | 268 |
| Chapter 7 UART | 270 |

| | | |
|------------|---|-----|
| 7.1 | Overview | 270 |
| 7.2 | Block Diagram..... | 270 |
| 7.3 | Function Description..... | 271 |
| 7.4 | Register Description | 273 |
| 7.5 | Interface Description | 290 |
| 7.6 | Application Notes..... | 291 |
| 7.6.1 | None FIFO Mode Transfer Flow | 291 |
| 7.6.2 | FIFO Mode Transfer Flow | 291 |
| 7.6.3 | Baud Rate Calculation | 292 |
| Chapter 8 | GPIO | 294 |
| 8.1 | Overview | 294 |
| 8.2 | Block Diagram | 294 |
| 8.3 | Function Description | 294 |
| 8.4 | Register Description..... | 296 |
| 8.5 | Interface Description | 299 |
| 8.6 | Application Notes | 300 |
| Chapter 9 | I2C Interface | 301 |
| 9.1 | Overview | 301 |
| 9.2 | Block Diagram | 301 |
| 9.3 | Function Description | 301 |
| 9.4 | Register Description | 304 |
| 9.5 | Interface Description | 312 |
| 9.6 | Application Notes | 313 |
| Chapter 10 | I2S/PCM Controller | 316 |
| 10.1 | Overview..... | 316 |
| 10.2 | Block Diagram | 317 |
| 10.3 | Function description..... | 317 |
| 10.4 | Register Description..... | 320 |
| 10.5 | Interface description | 330 |
| 10.6 | Application Notes..... | 332 |
| Chapter 11 | Serial Peripheral Interface (SPI) | 334 |
| 11.1 | Overview..... | 334 |
| 11.2 | Block Diagram | 334 |
| 11.3 | Function Description | 335 |
| 11.4 | Register Description | 337 |
| 11.5 | Interface Description | 347 |
| 11.6 | Application Notes..... | 347 |
| Chapter 12 | SPDIF transmitter | 349 |
| 12.1 | Overview..... | 349 |
| 12.2 | Block Diagram | 349 |
| 12.3 | Function description..... | 350 |
| 12.4 | Register description | 352 |
| 12.5 | Interface description | 361 |
| 12.6 | Application Notes..... | 362 |
| Chapter 13 | TSP(Transport Stream Processing Module) | 364 |
| 13.1 | Overview..... | 364 |
| 13.2 | Block Diagram | 364 |
| 13.3 | Function Description | 365 |
| 13.4 | Register Description | 367 |
| 13.5 | Application Notes..... | 409 |
| Chapter 14 | Temperature-Sensor ADC(TS-ADC) | 414 |
| 14.1 | Overview..... | 414 |

| | |
|--|-----|
| 14.2 Block Diagram | 414 |
| 14.3 Function Description | 414 |
| 14.4 Register description | 415 |
| 14.5 Application Notes..... | 421 |
| Chapter 15 GMAC Ethernet Interface..... | 424 |
| 15.1 Overview..... | 424 |
| 15.2 Block Diagram | 425 |
| 15.3 Function Description | 425 |
| 15.4 Register Description..... | 429 |
| 15.5 Interface Description..... | 478 |
| 15.6 Application Notes..... | 479 |
| Chapter 16 Smart Card Reader (SCR) | 491 |
| 16.1 Overview..... | 491 |
| 16.2 Block Diagram | 491 |
| 16.3 Function Description | 492 |
| 16.4 Register Description..... | 494 |
| 16.5 Interface Description..... | 507 |
| 16.6 Application Notes..... | 507 |
| Chapter 17 MACPHY | 509 |
| 17.1 Overview..... | 509 |
| 17.2 Block Diagram | 509 |
| 17.3 Function Description | 510 |

Figure Index

| | |
|--|-----|
| Fig. 5-1 Host Controller Block Diagram | 13 |
| Fig. 5-2 SD/MMC Card-Detect Signal | 16 |
| Fig. 5-3 Host Controller Command Path State Machine..... | 18 |
| Fig. 5-4 Host Controller Data Transmit State Machine | 20 |
| Fig. 5-5 Host Controller Data Receive State Machine | 22 |
| Fig. 5-6 Dual-Buffer Descriptor Structure | 27 |
| Fig. 5-7 Chain Descriptor Structure | 28 |
| Fig. 5-8 Descriptor Formats for 32-bit AHB Address Bus Width | 28 |
| Fig. 5-9 SD/MMC Card-Detect and Write-Protect | 65 |
| Fig. 5-10 SD/MMC Card Termination | 65 |
| Fig. 5-11 Host Controller Initialization Sequence | 67 |
| Fig. 5-12 Voltage Switching Command Flow Diagram | 76 |
| Fig. 5-13 ACMD41 Argument..... | 77 |
| Fig. 5-14 ACMD41 Response(R3)..... | 77 |
| Fig. 5-15 Voltage Switch Normal Scenario | 78 |
| Fig. 5-16 Voltage Switch Error Scenario | 79 |
| Fig. 5-17 CASES for eMMC 4.5 START bit | 81 |
| Fig. 5-18 Clock Generation Unit | 82 |
| Fig. 5-19 Card Detection Method 2..... | 85 |
| Fig. 5-20 Card Detection Method 4 | 86 |
| Fig. 5-21 USB HOST 2.0 Architecture..... | 206 |
| Fig. 5-22 HDMI TX Block Diagram | 213 |
| Fig. 5-23 HDMI Color Space Conversion Matrix Equations | 214 |
| Fig. 5-24 HDMI Audio Data Processing Diagram | 215 |
| Fig. 5-25 HDMI Audio Clock Regeneration Model | 216 |
| Fig. 5-26 Sample INNO HDMI2.0 TX PHY Block Diagram..... | 218 |
| Fig. 5-27 Power relative signals timing..... | 218 |
| Fig. 5-28 HDMI TX Block Diagram | 229 |
| Fig. 5-29 HDMI TX Block Diagram | 230 |
| Fig. 5-30 HDMI TX Block Diagram | 233 |
| Fig. 5-31 HDMI TX Block Diagram | 237 |
| Fig. 5-32 Pre-PLL Configuration for Main Typical Resolution | 238 |
| Fig. 5-1 hdcp22 Controller Block Diagram | 239 |
| Fig. 5-2 Transmitter Authentication | 241 |
| Fig. 5-3 ESM Image tool flow..... | 242 |
| Fig. 5-4 ESM software | 247 |
| Fig. 5-5 Host Library Layers | 249 |
| Fig. 5-6 The development tree overview | 250 |
| Fig. 5-7 PWM Block Diagram | 252 |
| Fig. 5-8 PWM Capture Mode | 253 |
| Fig. 5-9 PWM Continuous Left-aligned Output Mode | 253 |
| Fig. 5-10 PWM Continuous Center-aligned Output Mode | 253 |
| Fig. 5-11 PWM One-shot Center-aligned Output Mode | 254 |
| Fig. 5-12 UART Architecture | 270 |
| Fig. 5-13 UART Serial protocol | 271 |
| Fig. 5-14 IrDA 1.0 | 271 |
| Fig. 5-15 UART baud rate..... | 271 |
| Fig. 5-16 UART Auto flow control block diagram | 272 |
| Fig. 5-17 UART AUTO RTS TIMING | 273 |
| Fig. 5-18 UART AUTO CTS TIMING | 273 |
| Fig. 5-19 UART none fifo mode | 291 |
| Fig. 5-20 UART fifo mode..... | 292 |
| Fig. 5-21 UART clock generation | 292 |
| Fig. 5-22GPIO block diagram..... | 294 |
| Fig. 5-23 GPIO Interrupt RTL Block Diagram..... | 295 |

| | |
|--|-----|
| Fig. 5-24 I2C architecture | 301 |
| Fig. 5-25 I2C DATA Validity | 303 |
| Fig. 5-26 I2C Start and stop conditions..... | 303 |
| Fig. 5-27 I2C Acknowledge | 304 |
| Fig. 5-28 I2C byte transfer..... | 304 |
| Fig. 5-29 I2C Flow chat for transmit only mode | 313 |
| Fig. 5-30 I2C Flow chat for receive only mode | 314 |
| Fig. 5-31 I2C Flow chat for mix mode..... | 315 |
| Fig. 10-1 I2S/PCM controller (8 channel) Block Diagram..... | 317 |
| Fig. 10-2 I2S transmitter-master & receiver-slave condition..... | 317 |
| Fig. 10-3 I2S transmitter-slave& receiver-master condition..... | 318 |
| Fig. 10-4 I2S normal mode timing format | 318 |
| Fig. 10-5 I2S left justified mode timing format..... | 318 |
| Fig. 10-6 I2S right justified mode timing format..... | 318 |
| Fig. 10-7 PCM early mode timing format | 319 |
| Fig. 10-8 PCM late1 mode timing format | 319 |
| Fig. 10-9 PCM late2 mode timing format | 320 |
| Fig. 10-10 PCM late3 mode timing format | 320 |
| Fig. 10-11 I2S/PCM controller transmit operation flow chart..... | 332 |
| Fig. 10-12 I2S/PCM controller receive operation flow chart | 333 |
| Fig. 10-13 SPI Controller Block diagram..... | 335 |
| Fig. 10-14 SPI Master and Slave Interconnection | 335 |
| Fig. 10-15 SPI Format (SCPH=0 SCPOL=0) | 336 |
| Fig. 10-16 SPI Format (SCPH=0 SCPOL=1) | 336 |
| Fig. 10-17 SPI Format (SCPH=1 SCPOL=0) | 337 |
| Fig. 10-18 SPI Format (SCPH=1 SCPOL=1) | 337 |
| Fig. 10-19 SPI Master transfer flow diagram | 348 |
| Fig. 10-20 SPDIF transmitter Block Diagram | 349 |
| Fig. 10-21 SPDIF Frame Format..... | 350 |
| Fig. 10-22 SPDIF Sub-frame Format..... | 350 |
| Fig. 10-23 SPDIF Channel Coding | 351 |
| Fig. 10-24 SPDIF Preamble | 351 |
| Fig. 10-25 Format of Data-burst | 352 |
| Fig. 10-26 SPDIF transmitter operation flow chart | 362 |
| Fig. 10-27 SPI Slave transfer flow diagram..... | 363 |
| Fig. 1-28 TSP architecture | 365 |
| Fig. 1-29 Sync/Valid Serial Mode with Msb-Lsb Bit Ordering | 366 |
| Fig. 1-30 Sync/valid Parallel Mode..... | 366 |
| Fig. 1-31 Sync/Burst Parallel Mode | 366 |
| Fig. 1-32 Nosync/Valid Parallel Mode | 366 |
| Fig. 10-33 TS-ADC Controller Block Diagram | 414 |
| Fig. 10-34 tsadc timing diagram in bypass mode..... | 421 |
| Fig. 10-35 tsadc timing diagram in normal mode with tsadc_clk_sel = 1'b0 | 421 |
| Fig. 10-36 tsadc timing diagram in normal mode with tsadc_clk_sel = 1'b1 | 421 |
| Fig. 10-37GMACArchitecture | 425 |
| Fig. 10-38MAC Block Diagram | 425 |
| Fig. 10-39 RMII transmission bit ordering..... | 426 |
| Fig. 10-40Start of MII and RMII transmission in 100-Mbps mode | 426 |
| Fig. 10-41End of MII and RMII Transmission in 100-Mbps Mode..... | 426 |
| Fig. 10-42Start of MII and RMII Transmission in 10-Mbps Mode..... | 426 |
| Fig. 10-43End of MII and RMII Transmission in 10-Mbps Mode | 427 |
| Fig. 10-44 RMII receive bit ordering | 427 |
| Fig. 10-45 MDIO frame structure | 428 |
| Fig. 10-46Descriptor Ring and Chain Structure..... | 479 |
| Fig. 10-47Rx/Tx Descriptors definition | 480 |
| Fig. 10-48 RMII clock architecture when clock source from CRU | 488 |
| Fig. 10-49 RMII clock architecture when clock source from external OSC | 488 |

| | |
|--|-----|
| Fig. 10-50 RGMII clock architecture when clock source from CRU | 488 |
| Fig. 10-51 Wake-Up Frame Filter Register | 489 |
| Fig. 10-52 SCR Block Diagram..... | 491 |
| Fig. 10-53 Activation, Cold Reset and ATR | 493 |
| Fig. 10-54 Warm Reset and ATR | 494 |
| Fig. 10-55 Deactivation Sequence | 494 |
| Fig. 10-56 AFE Block Diagram | 509 |

Rockchip Confidential

Table Index

| | |
|--|-----|
| Table 5-1 Bits in Interrupt Status Register..... | 15 |
| Table 5-2 Auto-Stop Generation | 23 |
| Table 5-3 Non-data Transfer Commands and Requirements..... | 24 |
| Table 5-4 Bits in IDMAC DES0 Element | 28 |
| Table 5-5 Bits in IDMAC DES1 Element | 29 |
| Table 5-6 Bits in IDMAC DES2 Element | 29 |
| Table 5-7 Bits in IDMAC DES3 Element | 30 |
| Table 5-8 IOMUX Settings for SDMMC..... | 63 |
| Table 5-9 IOMUX Settings 1 for SDIO | 63 |
| Table 5-10 IOMUX Settings 2 for SDIO..... | 64 |
| Table 5-11 IOMUX Settings for eMMC | 64 |
| Table 5-12 Recommended Usage of use_hold_reg | 66 |
| Table 5-13 Command Settings for No-Data Command | 70 |
| Table 5-14 Command Setting for Single or Multiple-Block Read | 71 |
| Table 5-15 Command Settings for Single or Multiple-Block Write | 72 |
| Table 5-16 PBL and Watermark Levels | 81 |
| Table 5-17 Configuration for SDMMC Clock Generation | 82 |
| Table 5-18 Configuration for SDIO Clock Generation | 83 |
| Table 5-19 Configuration for eMMC Clock Generation | 83 |
| Table 5-20 Register for SDMMC Card Detection Method 3 | 86 |
| Table 5-21 USB OTG 2.0 Interface Description | 205 |
| Table 5-22USB HOST 2.0 Interface Description | 212 |
| Table 5-23 HDMI TX I2S 2 Channel Audio Sampling Frequency..... | 215 |
| Table 5-24 HDMI TX I2S 8 Channel Audio Sampling Frequency..... | 215 |
| Table 5-25 HDMI SPDIF Sampling Frequency at Each Video Format..... | 215 |
| Table 5-26 HDMI CTS and N table | 216 |
| Table 5-27 HDMI 3D structure table | 233 |
| Table 5-1 ESM average memory bandwidth usage | 240 |
| Table 5-2 ESM average memory bandwidth usage | 240 |
| Table 5-3 Secure Key for ESM..... | 241 |
| Table 5-4 ESM BSOD Output Mapping | 248 |
| Table 5-5 Requirements for DCP Tools..... | 248 |
| Table 5-6 Description of user adjustable parameters | 248 |
| Table 5-7 Common Platform Target Examples | 250 |
| Table 5-8 PWM Interface Description | 268 |
| Table 5-9 UART Interface Description..... | 290 |
| Table 5-10 UART baud rate configuration..... | 293 |
| Table 5-11 GPIO interface description | 299 |
| Table 5-12I2C Interface Description | 312 |
| Table 10-1 I2S Interface Description | 330 |
| Table 10-2 Interface Between I2S1 and ACODEC..... | 331 |
| Table 10-3 I2S Interface Between I2S2 and HDMI | 331 |
| Table 10-4 1SPI interface description..... | 347 |
| Table 10-5 SPDIF Interface Description | 361 |
| Table 10-6 Interface Between SPDIF And HDMI..... | 361 |
| Table 10-7 RMII Interface Description..... | 478 |
| Table 10-8 RGMII Interface Description..... | 478 |
| Table 10-9Receive Descriptor 0..... | 480 |
| Table 10-10Receive Descriptor 1 | 482 |
| Table 10-11Receive Descriptor 2 | 482 |
| Table 10-12Receive Descriptor 3 | 482 |
| Table 10-13 Transmit Descriptor 0..... | 483 |
| Table 10-14 Transmit Descriptor 1 | 484 |
| Table 10-15 Transmit Descriptor 2..... | 485 |
| Table 10-16 Transmit Descriptor 3..... | 485 |

| | |
|--|-----|
| Table 10-17NandC Interface Description | 507 |
| Table 10-18BAUDTUNE register..... | 508 |

Rockchip Confidential

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.

All copyright and patent rights referenced in this document belong to their respective owners and shall be subject to corresponding copyright and patent licensing requirements.

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 © 2015 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.

Chapter 1 Mobile Storage Host Controller

1.1 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 4.5) 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, SDIO, EMMC. The interface difference between these instances is shown in "Interface Description".

1.2 Block Diagram

The Host Controller consists of the following main functional blocks.

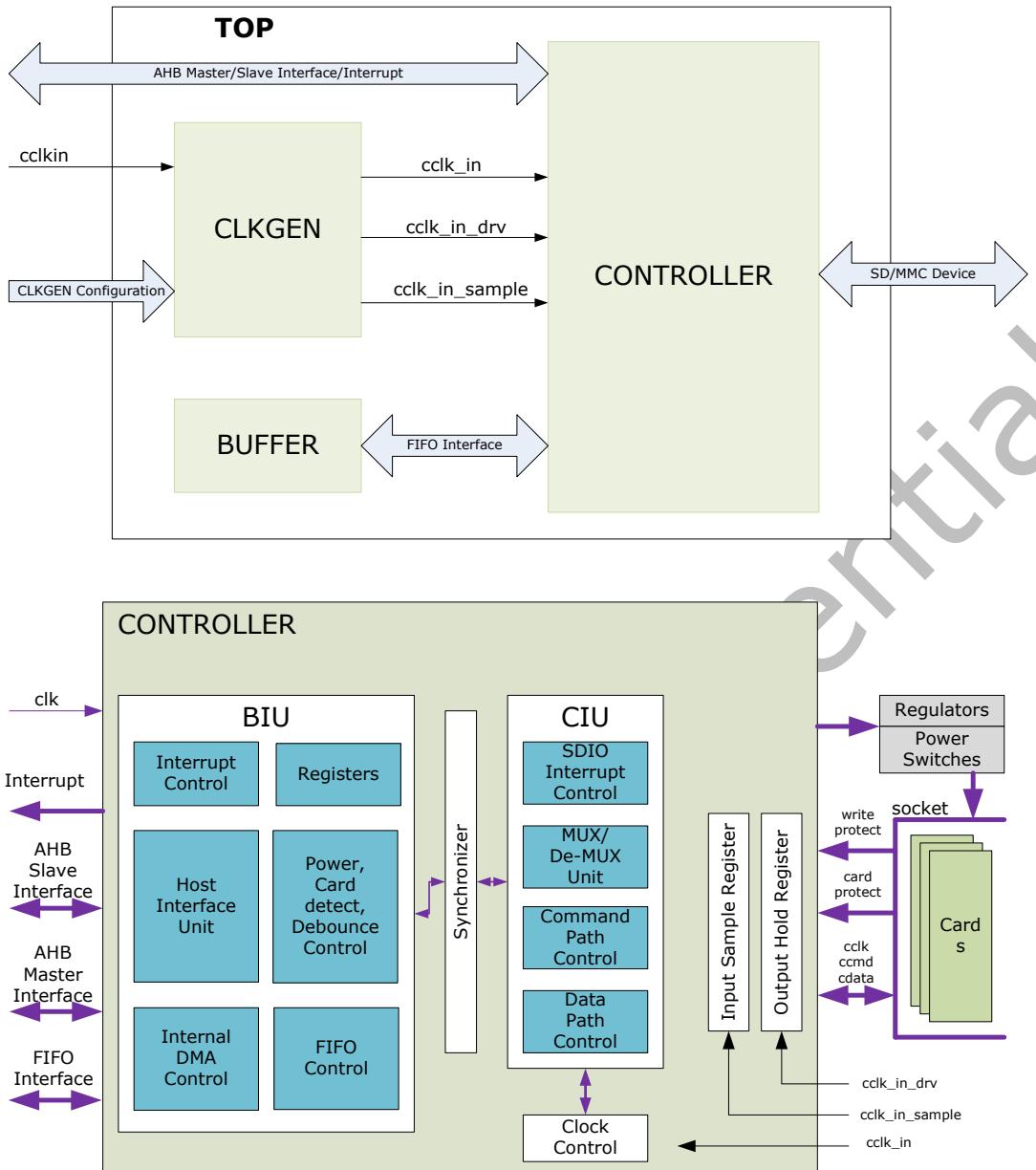


Fig. 5-1 Host Controller Block Diagram

- Clock Generate Unit(CLKGGEN): generates card interface clock `cclk_in/` `cclk_sample/cclk_drv` based on `cclkin` 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.

1.3 Function Description

1.3.1 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'ffff_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 5-1 Bits in Interrupt Status Register

| Bits | Interrupt | Description |
|-------------|--|--|
| 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) | 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. 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. 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. 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. |
| 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 | Interrupt set during write operation to card when FIFO |

| Bits | Interrupt | Description |
|------|--------------------------|---|
| | Request (TXDR) | level reaches less than or equal to Transmit-Threshold level. |
| 3 | Data Transfer Over (DTO) | Data transfer completed, even if there is Start Bit Error or CRC error. This bit is also set when "read data-timeout" occurs. <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> |
| 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">● Transmission bit != 0● Command index mismatch● End-bit != 1 |
| 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.

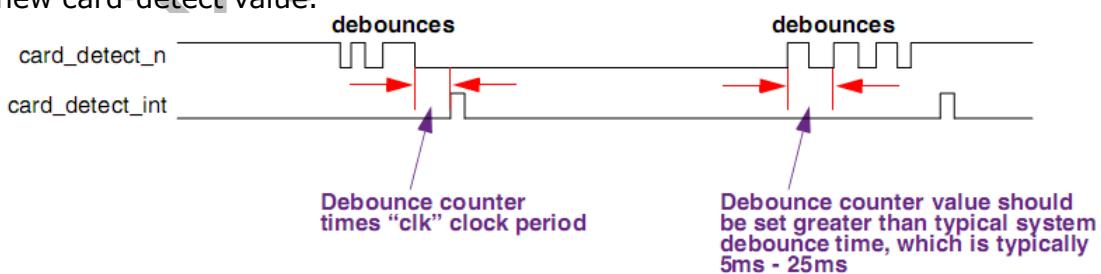


Fig. 5-2 SD/MMC Card-Detect Signal

1.3.2 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.

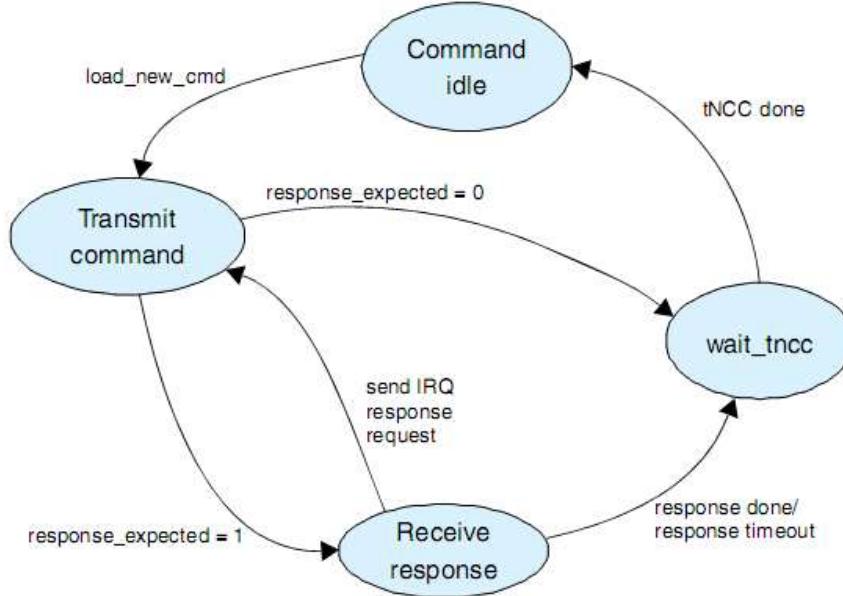


Fig. 5-3 Host Controller Command Path State Machine

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).

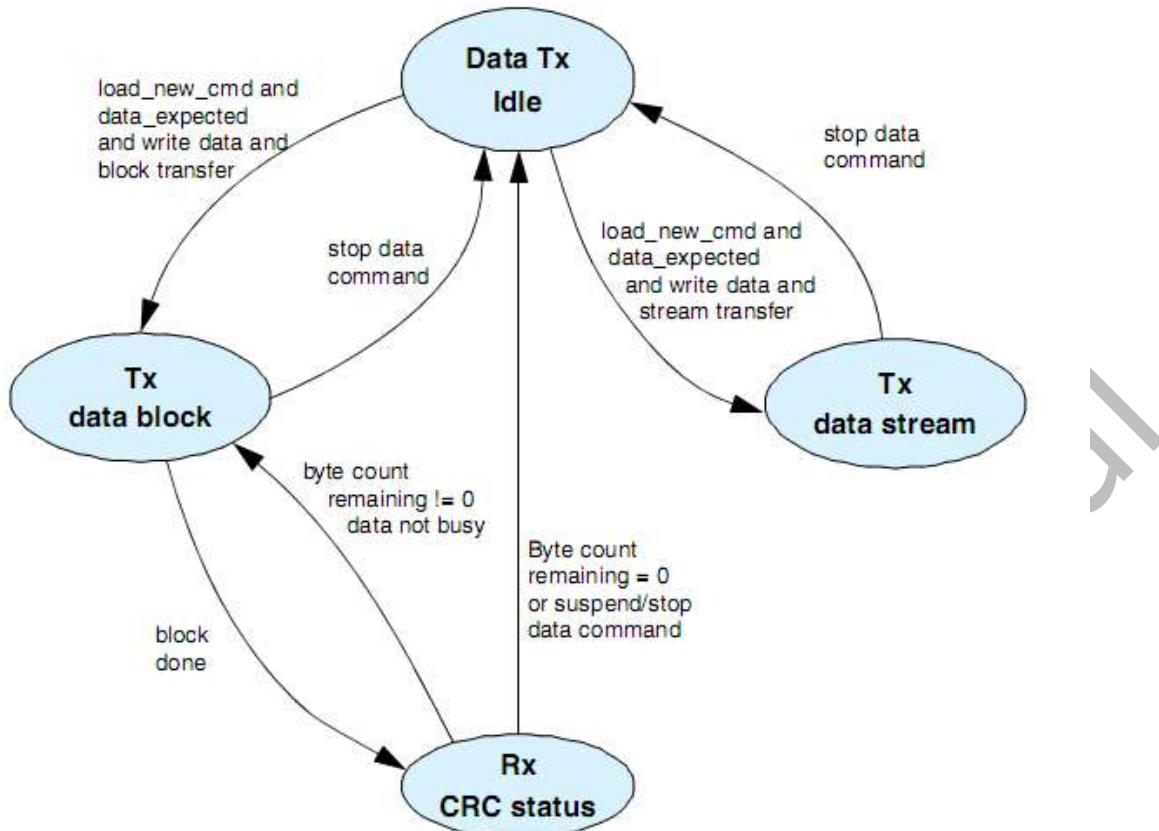


Fig. 5-4 Host Controller Data Transmit State Machine

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).

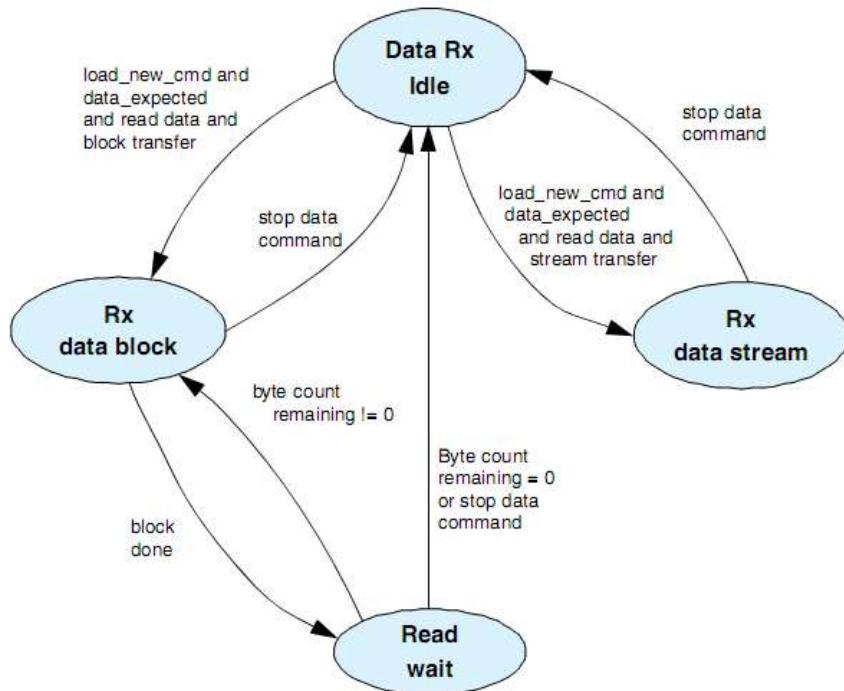


Fig. 5-5 Host Controller Data Receive State Machine

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 5-2 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 ^① | Pre-defined multiple block |
| MMC | Multiple-block write | 0 | No | Open-ended multiple block |
| MMC | Multiple-block write | >0 | Yes ^① | 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 |

^①: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 5-3 Non-data Transfer Commands and Requirements

| | CMD27 | CMD30 | CMD42 | ACMD13 | ACMD22 | ACMD51 |
|--|------------|------------------------|------------|------------|------------|------------|
| Command register programming | | | | | | |
| 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 |
| Command Argument register programming | | | | | | |
| | stuff bits | 32-bit writeprotect | stuff bits | stuff bits | stuff bits | stuff bits |

| | | | | | | | |
|--|----|---|------------------------|----|---|---|--|
| | | | dataaddress | | | | |
| Block Size register programming | | | | | | | |
| | 16 | 4 | Num_bytes ^① | 64 | 4 | 8 | |
| Byte Count register programming | | | | | | | |
| | 16 | 4 | Num_bytes ^① | 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.

1.3.3 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).

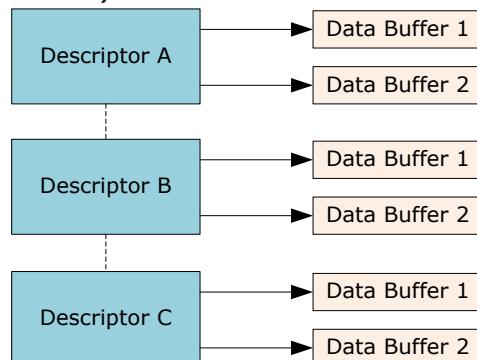


Fig. 5-6 Dual-Buffer Descriptor Structure

- Chain Structure – Each descriptor points to a unique buffer and the next descriptor.

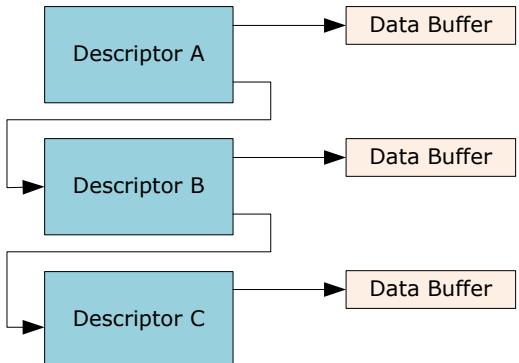


Fig. 5-7 Chain Descriptor Structure

- 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.

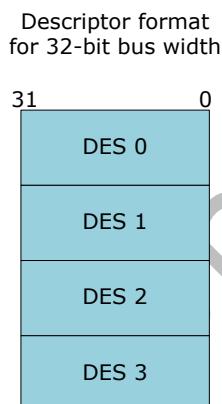


Fig. 5-8 Descriptor Formats for 32-bit AHB Address Bus Width

- The DES0 element in the IDMAC contains control and status information.

Table 5-4 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"> ➤ EBE: End Bit Error ➤ RTO: Response Time out ➤ RCRC: Response CRC ➤ SBE: Start Bit Error ➤ DRTO: Data Read Timeout ➤ DCRC: Data CRC for Receive ➤ RE: Response Error |
| 29:6 | Reserved | - |

| Bits | Name | Description |
|-------------|---------------------------------------|--|
| 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 5-5 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 5-6 Bits in IDMAC DES2 Element

| Bits | Name | Description |
|-------------|---------------|---|
| 31:26 | Reserved | |
| 25:13 | Buffer 2 Size | These bits indicate the second data buffer byte size. The buffer size |

| Bits | Name | Description |
|------|------------------------|---|
| | (BS2) | 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 5-7 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.

1.4 Register Description

1.4.1 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_TMOOUT | 0x0014 | W | 0xfffffff40 | Time-out register |
| SDMMC_CTYPE | 0x0018 | W | 0x00000000 | Card-type register |
| SDMMC_BLKSIZE | 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 |
| 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 | 0x00ffff | Card detect debounce register |
| SDMMC_USRID | 0x0068 | W | 0x07967797 | User ID register |

| Name | Offset | Size | Reset Value | Description |
|----------------------|--------|------|-------------|---|
| SDMMC_VERID | 0x006c | W | 0x5342270a | 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

1.4.2 Detail Register Description

SDMMC_CTRL

Address: Operational Base + offset (0x0000)

Control register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 |
| 11 | RW | 0x0 | ceata_device_interrupt_status 0: Interrupts not enabled in CE-ATA device (nIEN = 1 in ATA control register) 1: Interrupts are enabled in CE-ATA device (nIEN = 0 in ATA control register) 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. |
| 10 | RW | 0x0 | send_auto_stop_ccsd 0: Clear bit if DWC_mobile_storage does not reset the bit. 1: Send internally generated STOP after sending CCSD to CE-ATA device. 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) biin 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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</p> <p>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.</p> <p>Used in SDIO card suspend sequence.</p> |
| 7 | RW | 0x0 | <p>send_irq_response 0: no change 1: send auto IRQ response</p> <p>Bit automatically clears once response is sent.</p> <p>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> |
| 6 | RW | 0x0 | <p>read_wait 0: clear read wait 1: assert read wait</p> <p>For sending read-wait to SDIO cards</p> |
| 5 | RW | 0x0 | <p>dma_enable 0: disable DMA transfer mode 1: enable DMA transfer mode</p> <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 4 | RW | 0x0 | <p>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.</p> |
| 3 | RO | 0x0 | reserved |
| 2 | W1C | 0x0 | <p>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. The DMA should be reset before it switches from no-dma mode into dma mode.</p> |
| 1 | W1C | 0x0 | <p>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</p> |
| 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. 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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|-------------|
| 31:1 | RO | 0x0 | reserved |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:24 | RW | 0x00 | <p>clk_divider3 Clock divider-3 value. For example, value of 0 means divide by $2^0 = 0$ (no division, bypass), value of 1 means divide by $2^1 = 2$, value of "ff"means divide by $2^{255} = 510$, 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> |
| 23:16 | RW | 0x00 | <p>clk_divider2 Clock divider-2 value. Clock division is 2^n. For example, value of 0 means divide by $2^0 = 0$ (no division, bypass), value of 1 means divide by $2^1 = 2$, value of "ff"means divide by $2^{255} = 510$, 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> |
| 15:8 | RW | 0x00 | <p>clk_divider1 Clock divider-1 value. Clock division is 2^n. For example, value of 0 means divide by $2^0 = 0$ (no division, bypass), value of 1 means divide by $2^1 = 2$, value of "ff"means divide by $2^{255} = 510$, 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> |
| 7:0 | RW | 0x00 | <p>clk_divider0 Clock divider-0 value. Clock division is 2^n. For example, value of 0 means divide by $2^0 = 0$ (no division, bypass), value of 1 means divide by $2^1 = 2$, value of "ff"means divide by $2^{255} = 510$, and so on. In our design, the divider is 0 or 1 supported.</p> |

SDMMC_CLKSRC

Address: Operational Base + offset (0x000c)

SD Clock Source Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:2 | RO | 0x0 | reserved |
| 1:0 | RW | 0x0 | <p>clk_source</p> <p>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"> 00 –Clock divider 0 01 –Clock divider 1 10 –Clock divider 2 11 –Clock divider 3 <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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:17 | RO | 0x0 | reserved |
| 16 | RW | 0x0 | <p>cclk_low_power</p> <p>Low-power control for SD card clock and MMC card clock supported.</p> <p>0: non-low-power mode</p> <p>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</p> <p>Clock-enable control for SD card clock and MMC card clock supported.</p> <p>0: clock disabled</p> <p>1: clock enabled</p> |

SDMMC_TMOUT

Address: Operational Base + offset (0x0014)

Time-out register

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|----------------------------------|
| 31:16 | RO | 0x0 | reserved |
| 15:0 | RW | 0x0200 | <p>block_size Block size</p> |

SDMMC_BYTCNT

Address: Operational Base + offset (0x0020)

Byte-count register

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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> |
| 15:0 | RW | 0x0000 | <p>int_mask Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt.</p> <ul style="list-style-type: none"> 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) <p>/Volt_switch_int</p> <ul style="list-style-type: none"> 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_CMDARG

Address: Operational Base + offset (0x0028)

Command-argument register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RW | 0x00000000 | cmd_arg Value indicates command argument to be passed to card. |

SDMMC_CMD

Address: Operational Base + offset (0x002c)

Command register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31 | RW | 0x0 | start_cmd Start command. Once command is taken by CIU, bit is cleared. 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. Once command is sent and response is received from SD_MMC cards, Command Done bit is set in raw interrupt register. |
| 30 | RO | 0x0 | reserved |
| 29 | RW | 0x0 | 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 Note: 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. b. Set to 1'b0 for SDR50, SDR104, and DDR50 (with zero phase-shifted cclk_in_drv) c. Set to 1'b1 for SDR50, SDR104, and DDR50 (with non-zero phase-shifted cclk_in_drv) |
| 28 | RW | 0x0 | volt_switch Voltage switch bit. 0: no voltage switching 1: voltage switching enabled; must be set for CMD11 only |
| 27 | RW | 0x0 | boot_mode Boot Mode. 0: mandatory Boot operation 1: alternate Boot operation |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 26 | RW | 0x0 | 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. |
| 25 | RW | 0x0 | 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. |
| 24 | RW | 0x0 | 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. |
| 23 | RW | 0x0 | 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 If the command expects Command Completion Signal (CCS) from the CE-ATA device, the software should set this control bit. 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. |
| 22 | RW | 0x0 | 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 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. Maximum value of I/O transmission delay can be no less than 10 seconds. DWC_mobile_storage should not indicate read data timeout while waiting for data from CE-ATA device. |

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 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</p> <p>Following register values transferred into card clock domain: CLKDIV, CLRSRC, CLKENA.</p> <p>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.</p> <p>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</p> <p>Card number in use. Represents physical slot number of card being accessed. In MMC-Ver3.3-only mode, up to 30 cards are supported;</p> <p>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.</p> <p>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> |
| 15 | RW | 0x0 | <p>send_initialization</p> <p>0: do not send initialization sequence (80 clocks of 1) before sending this command</p> <p>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> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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> |
| 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</p> <p>When set, SDMMC Controller sends stop command to SD_MMC cards at end of data transfer.</p> <ul style="list-style-type: none"> * when send_auto_stop bit should be set, since some data transfers do not need explicit stop commands * open-ended transfers that software should explicitly send to stop command <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.</p> <p>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</p> <p>Don't care if no data expected.</p> |
| 10 | RW | 0x0 | <p>wr 0: read from card 1: write to card</p> <p>Don't care if no data expected from card.</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 9 | RW | 0x0 | data_expected 0: no data transfer expected (read/write) 1: data transfer expected (read/write) |
| 8 | RW | 0x0 | 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 |
| 7 | RW | 0x0 | response_length 0: short response expected from card 1: long response expected from card |
| 6 | RW | 0x0 | response_expect 0: no response expected from card 1: response expected from card |
| 5:0 | RW | 0x00 | cmd_index Command index |

SDMMC_RESP0

Address: Operational Base + offset (0x0030)

Response-0 register

| Bit | Attr | Reset Value | Description |
|------|------|-------------|------------------------------------|
| 31:0 | RO | 0x00000000 | response0 Bit[31:0] of response |

SDMMC_RESP1

Address: Operational Base + offset (0x0034)

Response-1 register

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RO | 0x00000000 | response2 Bit[95:64] of long response |

SDMMC_RESP3

Address: Operational Base + offset (0x003c)

Response-3 register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RO | 0x00000000 | response3 Bit[127:96] of long response |

SDMMC_MINTSTS

Address: Operational Base + offset (0x0040)

Masked interrupt-status register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:25 | RO | 0x0 | reserved |
| 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 |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 15:0 | RO | 0x0000 | <p>int_status</p> <p>Interrupt enabled only if corresponding bit in interrupt mask register is set.</p> <p>bit 15: End-bit error (read)/Write no CRC (EBE)</p> <p>bit 14: Auto command done (ACD)</p> <p>bit 13: Start-bit error (SBE)</p> <p>bit 12: Hardware locked write error (HLE)</p> <p>bit 11: FIFO underrun/overrun error (FRUN)</p> <p>bit 10: Data starvation-by-host timeout (HTO) /Volt_switch_int</p> <p>bit 9: Data read timeout (DRTO)</p> <p>bit 8: Response timeout (RTO)</p> <p>bit 7: Data CRC error (DCRC)</p> <p>bit 6: Response CRC error (RCRC)</p> <p>bit 5: Receive FIFO data request (RXDR)</p> <p>bit 4: Transmit FIFO data request (TXDR)</p> <p>bit 3: Data transfer over (DTO)</p> <p>bit 2: Command done (CD)</p> <p>bit 1: Response error (RE)</p> <p>bit 0: Card detect (CD)</p> |

SDMMC_RINTSTS

Address: Operational Base + offset (0x0044)

Raw interrupt-status register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:25 | RO | 0x0 | reserved |
| 24 | RO | 0x0 | <p>sdio_interrupt</p> <p>Interrupt from SDIO card; Writes to these bits clear them. Value of 1 clears bit and 0 leaves bit intact.</p> <p>0: no SDIO interrupt from card</p> <p>1: SDIO interrupt from card</p> |
| 23:17 | RO | 0x0 | reserved |
| 16 | RW | 0x0 | <p>data_nobusy_int_status</p> <p>Data no busy interrupt status</p> |

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 15:0 | RO | 0x0000 | <p>int_status</p> <p>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.</p> <p>bit 15: End-bit error (read)/Write no CRC (EBE)</p> <p>bit 14: Auto command done (ACD)</p> <p>bit 13: Start-bit error (SBE)</p> <p>bit 12: Hardware locked write error (HLE)</p> <p>bit 11: FIFO underrun/overrun error (FRUN)</p> <p>bit 10: Data starvation-by-host timeout (HTO)</p> <p>/Volt_switch_int</p> <p>bit 9: Data read timeout (DRTO)</p> <p>bit 8: Response timeout (RTO)</p> <p>bit 7: Data CRC error (DCRC)</p> <p>bit 6: Response CRC error (RCRC)</p> <p>bit 5: Receive FIFO data request (RXDR)</p> <p>bit 4: Transmit FIFO data request (TXDR)</p> <p>bit 3: Data transfer over (DTO)</p> <p>bit 2: Command done (CD)</p> <p>bit 1: Response error (RE)</p> <p>bit 0: Card detect (CD)</p> |

SDMMC_STATUS

Address: Operational Base + offset (0x0048)

Status register

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 8 | RO | 0x0 | <p>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</p> |
| 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. 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"> * Bit 16 –Wait for CCS * Bit 17 –Send CCSD * Bit 18 –Boot Mode <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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. 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"> MSize = 1, TX_WMARK = 1-15 MSize = 4, TX_WMark = 8 MSize = 4, TX_WMark = 4 MSize = 4, TX_WMark = 12 MSize = 8, TX_WMark = 8 MSize = 8, TX_WMark = 4 <p>Allowed combinations for MSize and RX_WMark are:</p> <ul style="list-style-type: none"> MSize = 1, RX_WMARK = 0-14 MSize = 4, RX_WMark = 3 MSize = 4, RX_WMark = 7 MSize = 4, RX_WMark = 11 MSize = 8, RX_WMark = 7 <p>Recommended:</p> <ul style="list-style-type: none"> MSize = 8, TX_WMark = 8, RX_WMark = 7 |

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 27:16 | RW | 0x000 | <p>rx_wmark FIFO threshold watermark level when receiving data to card.</p> <p>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.</p> <p>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 <= FIFO_DEPTH-2</p> <p>Recommended: (FIFO_DEPTH/2) - 1; (means greater than (FIFO_DEPTH/2) - 1)</p> <p>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 |

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 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. 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. 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 >= 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

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

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

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:0 | RO | 0x00000000 | <p>trans_card_byte_count Number of bytes transferred by CIU unit to card. 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. 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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:0 | RO | 0x00000000 | <p>trans_fifo_byte_count Number of bytes transferred between Host/DMA memory and BIU FIFO. 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

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 31:24 | RO | 0x0 | reserved |
| 23:0 | RW | 0xfffffff | 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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 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_VRID

Address: Operational Base + offset (0x006c)

Version ID register

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

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 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"> [0]: CARD_TYPE <ul style="list-style-type: none"> 0: MMC_ONLY 1: SD_MMC [5:1]: NUM_CARDS - 1 [6]: H_BUS_TYPE <ul style="list-style-type: none"> 0: APB 1: AHB [9:7]: H_DATA_WIDTH <ul style="list-style-type: none"> 000: 16 bits 001: 32 bits 010: 64 bits others: reserved [15:10]: H_ADDR_WIDTH <ul style="list-style-type: none"> 0 to 7: reserved 8: 9 bits 9: 10 bits ... 31: 32 bits 32 to 63: reserved [17:16]: DMA_INTERFACE <ul style="list-style-type: none"> 00: none 01: DW_DMA 10: GENERIC_DMA 11: NON-DW-DMA [20:18]: GE_DMA_DATA_WIDTH <ul style="list-style-type: none"> 000: 16 bits 001: 32 bits 010: 64 bits others: reserved [21]: FIFO_RAM_INSIDE <ul style="list-style-type: none"> 0: outside 1: inside [22]: IMPLEMENT_HOLD_REG <ul style="list-style-type: none"> 0: no hold register 1: hold register [23]: SET_CLK_FALSE_PATH <ul style="list-style-type: none"> 0: no false path 1: false path set [25:24]: NUM_CLK_DIVIDER-1 [26]: AREA_OPTIMIZED <ul style="list-style-type: none"> 0: no area optimization 1: Area optimization |

SDMMC_UHS_REG

Address: Operational Base + offset (0x0074)

UHS-1 register

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

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

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 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 FIFO TH register. In order to change this value, write the required value to FIFO TH register. This is an encode value as follows.</p> <ul style="list-style-type: none"> 000 – 1 transfers 001 – 4 transfers 010 – 8 transfers 011 – 16 transfers 100 – 32 transfers 101 – 64 transfers 110 – 128 transfers 111 – 256 transfers <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. |
| 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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:17 | RO | 0x0 | reserved |
| 16:13 | RO | 0x0 | FSM DMAC FSM present state. 0: DMA_IDLE 1: DMA_SUSPEND 2: DESC_RD 3: DESC_CHK 4: DMA_RD_REQ_WAI 5: DMA_WR_REQ_WAI 6: DMA_RD 7: DMA_WR 8: DESC_CLOSE |
| 12:10 | RO | 0x0 | EB 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. 1: Host Abort received during transmission 2: Host Abort received during reception Others: Reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 9 | RW | 0x0 | AIS Abnormal Interrupt Summary. Logical OR of the following: IDSTS[2] Fatal Bus Interrupt IDSTS[4] DU bit Interrupt Only unmasked bits affect this bit. 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. |
| 8 | RW | 0x0 | NIS Normal Interrupt Summary. Logical OR of the following: IDSTS[0] Transmit Interrupt IDSTS[1] Receive Interrupt Only unmasked bits affect this bit. 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. |
| 7:6 | RO | 0x0 | reserved |
| 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"> ■ EBE –End Bit Error ■ RTO –Response Timeout/Boot Ack Timeout ■ RCRC –Response CRC ■ SBE –Start Bit Error ■ DRTO –Data Read Timeout/BDS timeout ■ DCRC –Data CRC for Receive ■ RE –Response Error 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 |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:10 | RO | 0x0 | reserved |
| 9 | RW | 0x0 | AI Abnormal Interrupt Summary Enable. When set, an abnormal interrupt is enabled. This bit enables the following bits: ■ IDINTEN[2] Fatal Bus Error Interrupt ■ IDINTEN[4] DU Interrupt |
| 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: ■ IDINTEN[0] Transmit Interrupt ■ IDINTEN[1] Receive Interrupt |
| 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 |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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_BUFAADDR

Address: Operational Base + offset (0x0098)

Current Buffer Descriptor Address Register

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:28 | RO | 0x0 | reserved |
| 27:16 | RW | 0x000 | CardRdThreshold Card Read Threshold size |

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 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. |
| 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

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|-------------|
| 31:0 | RW | 0x00000000 | Reserved |

SDMMC_EMMC_DDR_REG

Address: Operational Base + offset (0x010c)

eMMC 4.5 DDR START Bit Detection Control Register

| Bit | Attr | Reset Value | Description |
|------|------|-------------|-------------|
| 31:1 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 0 | RW | 0x0 | <p>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.</p> |

SDMMC_ENABLE_SHIFT

Address: Operational Base + offset (0x0110)

Enable Phase Shift Register

| Bit | Attr | Reset Value | Description |
|------|------|-------------|-------------|
| 31:0 | RW | 0x00000000 | Reserved |

SDMMC_FIFO_BASE

Address: Operational Base + offset (0x0200)

FIFO Base Address

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--------------------------------------|
| 31:0 | RW | 0x00000000 | fifo_base_addr FIFO base address. |

1.5 Interface Description

The interface and IOMUX setting for SDMMC, SDIO, EMMC are shown as follows.

1.5.1 SDMMC IOMUX

Table 5-8 IOMUX Settings for SDMMC

| Module Pin | Direction | Pad Name | IOMUX Setting |
|----------------|-----------|------------------------------|-------------------------------|
| sdmmc_cclk | O | IO_SDMMCClkout_GPIO1c0 | GRF_GPIO1C_IOMUX[1:0]=2'b01 |
| sdmmc_ccmd | I/O | IO_SDMMCCmd_GPIO1b7 | GRF_GPIO1B_IOMUX[15:14]=2'b01 |
| sdmmc_cdata0 | I/O | IO_SDMMCd0_UART2sout_GPIO1c2 | GRF_GPIO1C_IOMUX[5:4]=2'b01 |
| sdmmc_cdata1 | I/O | IO_SDMMCd1_UART2sin_GPIO1c3 | GRF_GPIO1C_IOMUX[7:6]=2'b01 |
| sdmmc_cdata2 | I/O | IO_SDMMCd2_JTAGtck_GPIO1c4 | GRF_GPIO1C_IOMUX[9:8]=2'b01 |
| sdmmc_cdata3 | I/O | IO_SDMMCd3_JTAGtms_GPIO1c5 | GRF_GPIO1C_IOMUX[11:10]=2'b01 |
| sdmmc_cdetectn | I | IO_SDMMCdetn_GPIO1c1 | GRF_GPIO1C_IOMUX[3:2]=2'b01 |
| sdmmc_wprt | I | IO_SDMMCwprt_GPIO1a7 | GRF_GPIO1A_IOMUX[15:14]=2'b01 |
| sdmmc_pwren | O | IO_SDMMCPwren_GPIO1b6 | GRF_GPIO1B_IOMUX[13:12]=2'b01 |

1.5.2 SDIO IOMUX

There are two groups of IOMUX setting for SDIO.

When GRF_CON_IOMUX[4] is 0, the "IOMUX Settings 1" is active; otherwise, "IOMUX Settings 2" is active.

Table 5-9 IOMUX Settings 1 for SDIO

| Module Pin | Direction | Pad Name | IOMUX Setting |
|-------------|-----------|----------------------------|-------------------------------|
| sdio_cclk | O | IO_SDIOClkout_GPIO1a0 | GRF_GPIO1A_IOMUX[1:0]=2'b01 |
| sdio_ccmd | I/O | IO_I2C1sda_SDIOcmd_GPIO0a3 | GRF_GPIO0A_IOMUX[7:6]=2'b01 |
| sdio_cdata0 | I/O | IO_SDIOD0_GPIO1a1 | GRF_GPIO1A_IOMUX[3:2]=2'b01 |
| sdio_cdata1 | I/O | IO_SDIOD1_I2Sdio1_GPIO1a2 | GRF_GPIO1A_IOMUX[5:4]=2'b01 |
| sdio_cdata2 | I/O | IO_SDIOD2_I2Sdio2_GPIO1a4 | GRF_GPIO1A_IOMUX[9:8]=2'b01 |
| sdio_cdata3 | I/O | IO_SDIOD3_I2Sdio3_GPIO1a5 | GRF_GPIO1A_IOMUX[11:10]=2'b01 |
| sdio_pwren | O | IO_SDIOpwren_PWM11_GPIO0d6 | GRF_GPIO0D_IOMUX[13:12]=2'b01 |

Table 5-10 IOMUX Settings 2 for SDIO

| Module Pin | Direction | Pad Name | IOMUX Setting |
|-------------|-----------|---------------------|-------------------------------|
| sdio_cclk | O | IO_SDIO1clk_GPIO3a0 | GRF_GPIO3A_IOMUX[1:0]=2'b01 |
| sdio_ccmd | I/O | IO_SDIO1cmd_GPIO3a1 | GRF_GPIO3A_IOMUX[3:2]=2'b01 |
| sdio_cdata0 | I/O | IO_SDIO1d0_GPIO3a2 | GRF_GPIO3A_IOMUX[5:4]=2'b01 |
| sdio_cdata1 | I/O | IO_SDIO1d1_GPIO3a3 | GRF_GPIO3A_IOMUX[7:6]=2'b01 |
| sdio_cdata2 | I/O | IO_SDIO1d2_GPIO3a4 | GRF_GPIO3A_IOMUX[9:8]=2'b01 |
| sdio_cdata3 | I/O | IO_SDIO1d3_GPIO3a5 | GRF_GPIO3A_IOMUX[11:10]=2'b01 |

1.5.3 eMMC IOMUX

Table 5-11 IOMUX Settings for eMMC

| Module Pin | Direction | Pad Name | IOMUX Setting |
|-------------|-----------|--------------------------------|--|
| emmc_cclk | O | IO_NANDdqs_EMMCclkout_GPIO2a7 | GRF_GPIO2A_IOMUX[15:14]=2'b10 |
| emmc_ccmd | I/O | IO_NANDcs2_EMMCCmd_GPIO1c6 | GRF_GPIO1C_IOMUX[13:12]=2'b10 & GRF_CON_IOMUX[7]=1'b0 |
| | | IO_NANDrdy_EMMC1cmd_GPIO2a4 | GRF_GPIO2A_IOMUX[9:8]=2'b10 & GRF_CON_IOMUX[7]=1'b1 |
| emmc_cdata0 | I/O | IO_NANDd0_EMMCD0_GPIO1d0 | GRF_GPIO1D_IOMUX[1:0]=2'b10 |
| emmc_cdata1 | I/O | IO_NANDd1_EMMCD1_GPIO1d1 | GRF_GPIO1D_IOMUX[3:2]=2'b10 |
| emmc_cdata2 | I/O | IO_NANDd2_EMMCD2_GPIO1d2 | GRF_GPIO1D_IOMUX[5:4]=2'b10 |
| emmc_cdata3 | I/O | IO_NANDd3_EMMCD3_GPIO1d3 | GRF_GPIO1D_IOMUX[7:6]=2'b10 |
| emmc_cdata4 | I/O | IO_NANDd4_EMMCD4_GPIO1d4 | GRF_GPIO1D_IOMUX[9:8]=2'b10 |
| emmc_cdata5 | I/O | IO_NANDd5_EMMCD5_GPIO1d5 | GRF_GPIO1D_IOMUX[11:10]=2'b10 |
| emmc_cdata6 | I/O | IO_NANDd6_EMMCD6_GPIO1d6 | GRF_GPIO1D_IOMUX[13:12]=2'b10 |
| emmc_cdata7 | I/O | IO_NANDd7_EMMCD7_GPIO1d7 | GRF_GPIO1D_IOMUX[15:14]=2'b10 |
| emmc_rstn | O | IO_NANDcs3_EMMCrstnout_GPIO1c7 | GRF_GPIO1C_IOMUX[15:14]=2'b10 |
| emmc_pwren | O | IO_NANDwp_EMMCPwren_GPIO2a5 | GRF_GPIO2A_IOMUX[11:10]=2'b10 |

Notes: Direction: **I**- Input, **O**- Output, **I/O**- Input/Output

1.6 Application Notes

1.6.1 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.

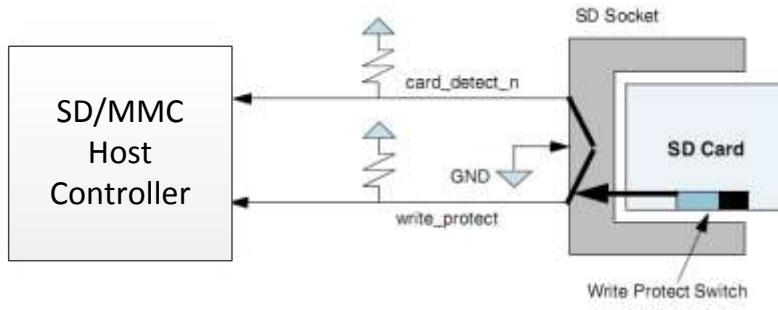


Fig. 5-9 SD/MMC Card-Detect and Write-Protect

1.6.2 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.

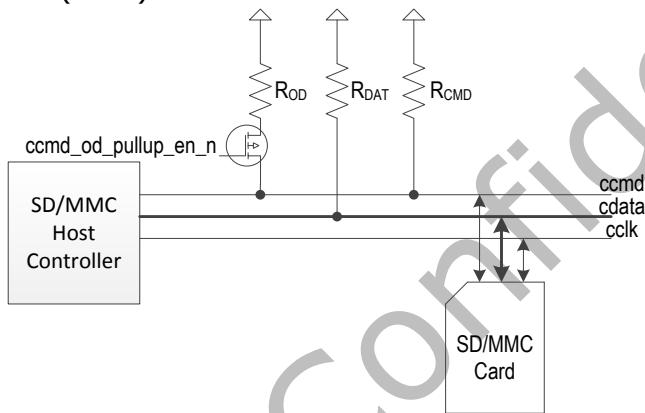


Fig. 5-10 SD/MMC Card Termination

1. Rcmd and Rod 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_{OL} 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:

$$\begin{aligned} 2.2 \text{ RC} &= \text{rise-time} = 1/400\text{KHz} \\ R &= 1/(2.2 * C * 100\text{KHz}) \\ &= 1/(2.2 * 20 * 10^{12} * 400 * 10^3) \\ &= 1/(1.76 * 10^{-5}) \end{aligned}$$

= 56.8K

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.

1.6.3 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 5-12 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.

1.6.4 Programming Sequence

1. Initialization

Following figure illustrates the initialization flow.

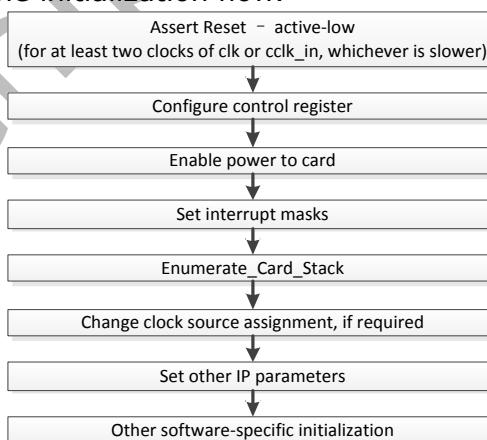


Fig. 5-11 Host Controller Initialization Sequence

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 5-13 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 |
| User-selectable | | |
| 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 5-14 Command Setting for Single or Multiple-Block Read

| 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 |
| 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 |
| User-selectable | | |
| 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 5-15 Command Settings for Single or Multiple-Block Write

| 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 |
| 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 |
| User-selectable | | |
| 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.

1.6.5 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.

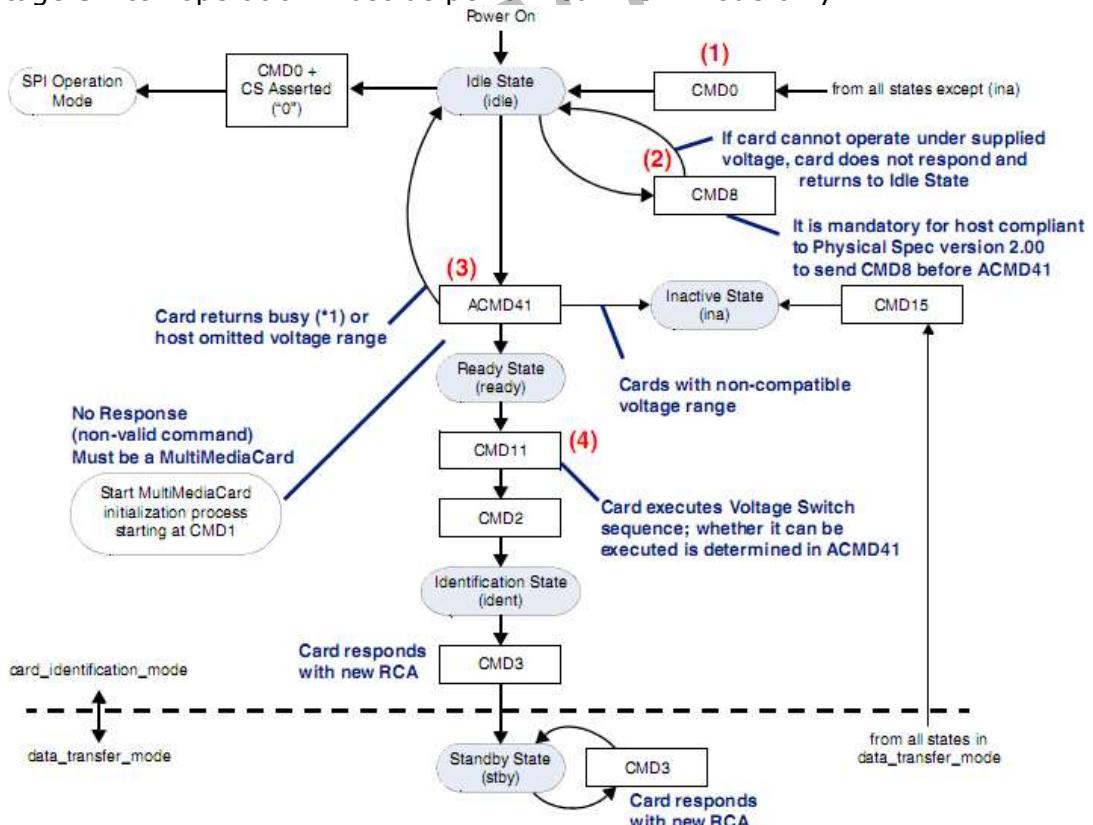


Fig. 5-12 Voltage Switching Command Flow Diagram

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.

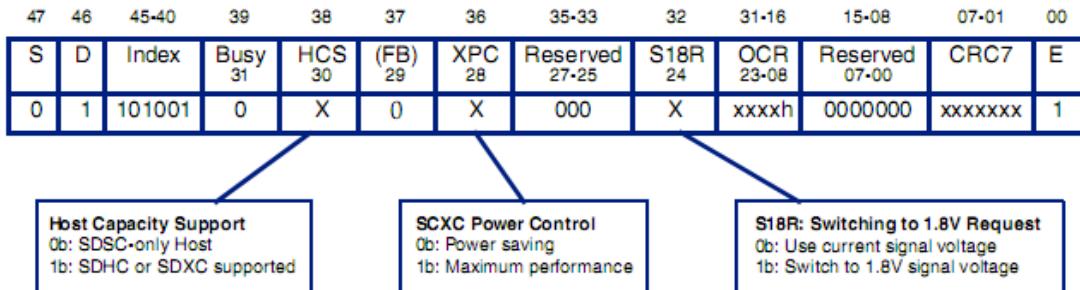


Fig. 5-13 ACMD41 Argument

- 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.

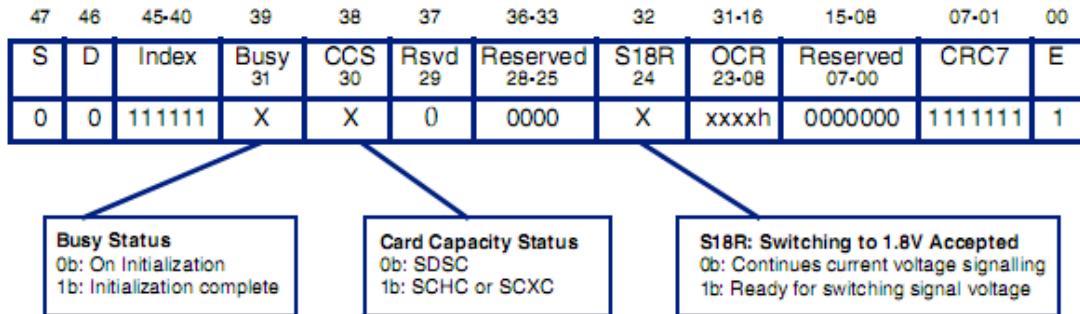


Fig. 5-14 ACMD41 Response(R3)

- 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

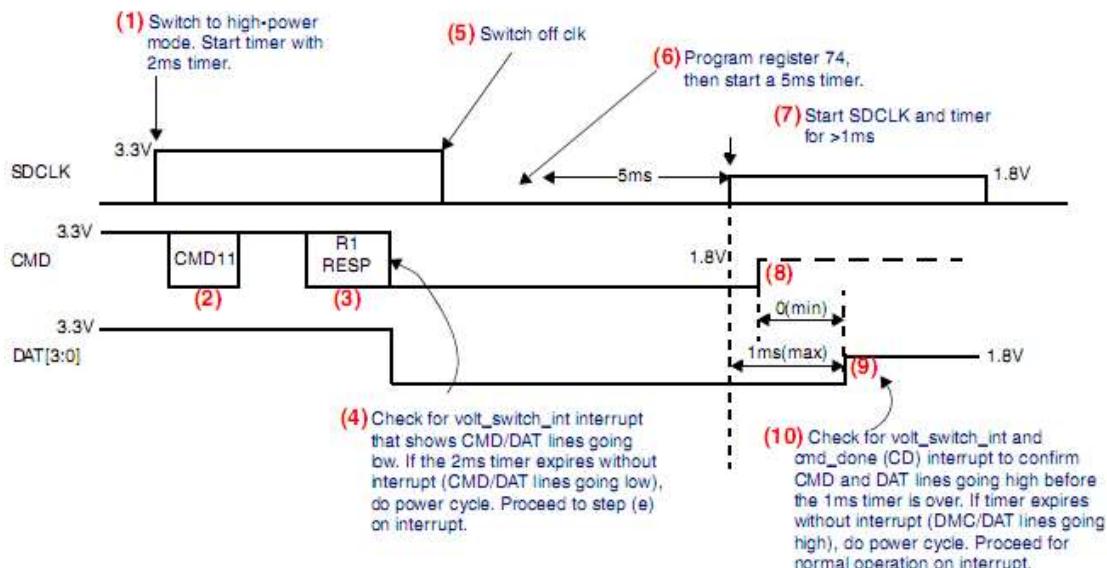


Fig. 5-15 Voltage Switch Normal Scenario

- 1) 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
- 2) 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".
- 3) The card returns R1 response; the host controller does not generate cmd_done interrupt on receiving R1 response.
- 4) 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.

- 5) Program the CLKENA, cclk_enable register, with 0 for the corresponding card; the host stops supplying SDCLK.
- 6) 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.
- 7) 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

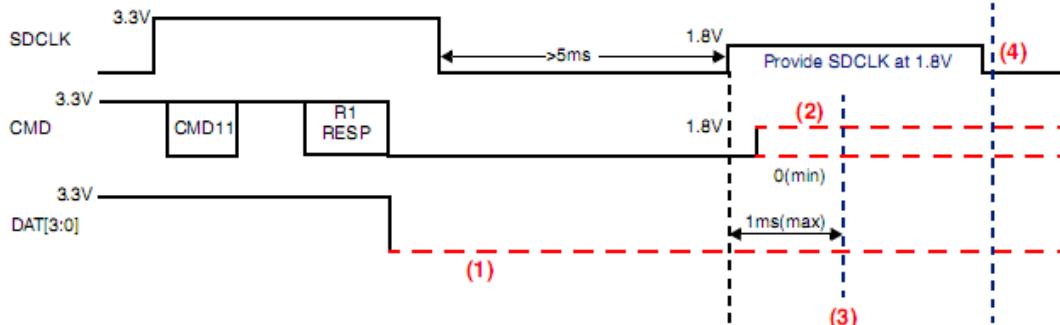


Fig. 5-16 Voltage Switch Error Scenario

- 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.

1.6.6 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.

1.6.7 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.

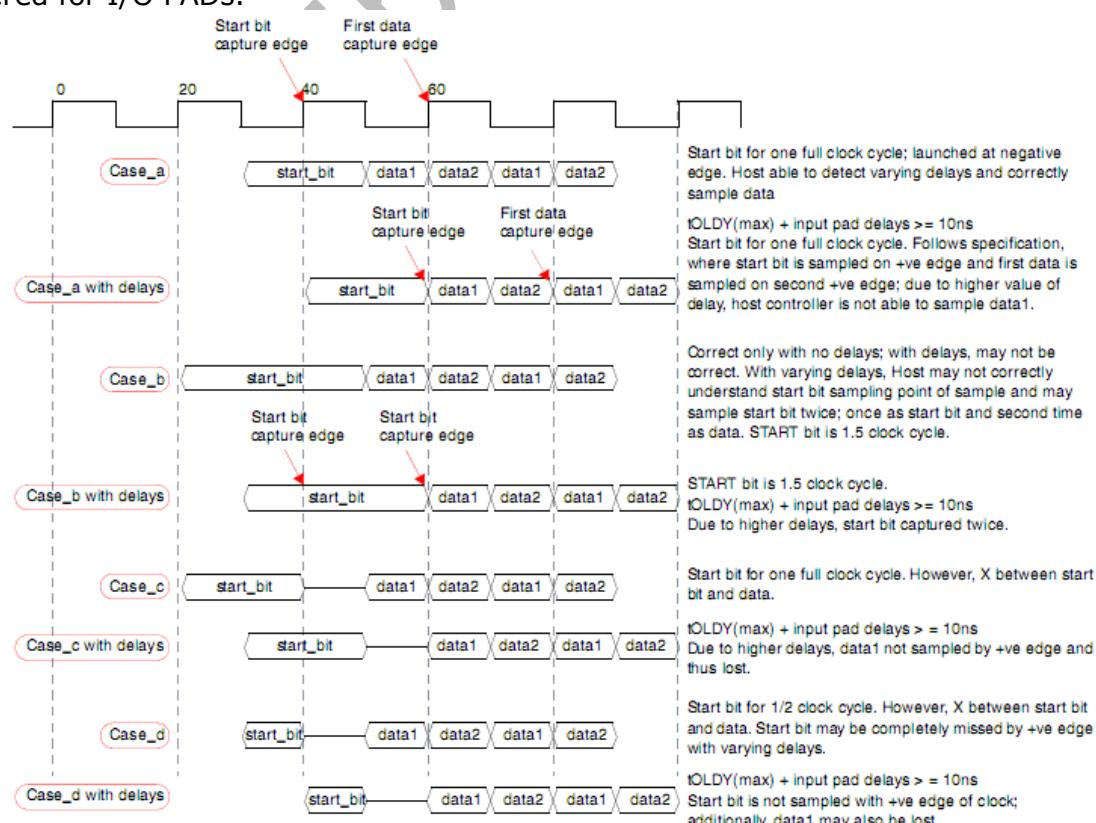


Fig. 5-17 CASES for eMMC 4.5 START bit

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.

1.6.8 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 μ s or cclk_in period, whichever is greater
- 6) After a minimum of 1 μ 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 μ 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.

1.6.9 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 5-16 PBL and Watermark Levels

| PBL (Number of transfers) | Tx/Rx Watermark Value |
|---------------------------|-----------------------|
|---------------------------|-----------------------|

| 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 |

1.6.10 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.

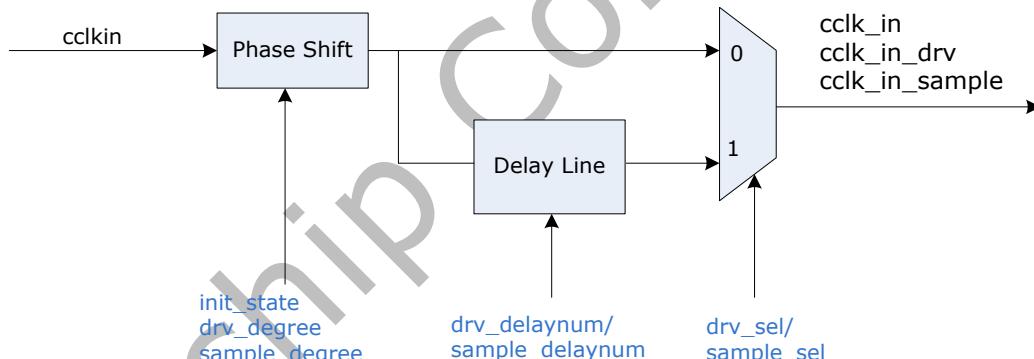


Fig. 5-18 Clock Generation Unit

The control signals for different Host Controller instance are shown as follows:

Table 5-17 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 |

| Signal Name | Source | Default | Description |
|-----------------------|----------------------|----------------|---|
| sample_degree [1:0] | CRU_SDMMC_CON1[2:1] | 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[10:3] | 0 | Element number in delay line for cclk_in_sample |
| sample_sel | CRU_SDMMC_CON1[11] | 0 | cclk_in_sample source selection: 0: use clock after phase_shift 1: use clock after phase_shift and delay line |

Table 5-18 Configuration for SDIO Clock Generation

| Signal Name | Source | Default | Description |
|-----------------------|---------------------|----------------|---|
| init_state | CRU_SDIO_CON0[0] | 0 | Soft initial state for phase shift. |
| drv_degree [1:0] | CRU_SDIO_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_SDIO_CON0[10:3] | 0 | Element number in delay line for cclk_in_drv |
| drv_sel | CRU_SDIO_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_SDIO_CON1[2:1] | 0 | Phase shift for cclk_in_sample. 0: 0-degree 1: 90-degree 2: 180-degree 3: 270-degree |
| sample_delaynum [7:0] | CRU_SDIO_CON1[10:3] | 0 | Element number in delay line for cclk_in_sample |
| sample_sel | CRU_SDIO_CON1[11] | 0 | cclk_in_sample source selection: 0: use clock after phase_shift 1: use clock after phase_shift and delay line |

Table 5-19 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 | CRU_EMMC_CON0[2:1] | 2 | Phase shift for cclk_in_drv. |

| Signal Name | Source | Default | Description |
|-----------------------|---------------------|----------------|---|
| [1:0] | | | 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[2:1] | 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[10:3] | 0 | Element number in delay line for cclk_in_sample |
| sample_sel | CRU_EMMC_CON1[11] | 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.

1.6.11 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.

1.6.12 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.

1.6.13 Card Detection Method

There are many methods for SDMMC/SDIO card detection.

- (1) Method1: Using CDETECT register, which is value on card_detect_n input port. 0 represents presence of card. Commonly for SDMMC/SDIO.
- (2) Method2: Using card detection unit, outputting host interrupt. 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/SDIO.

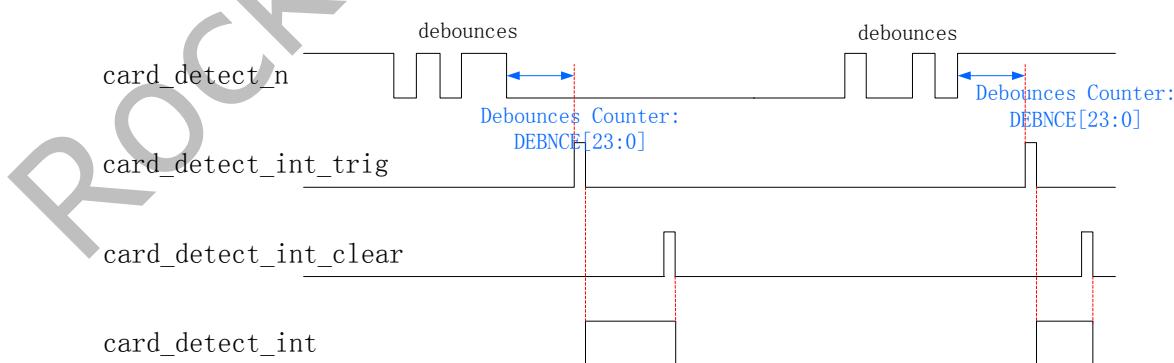


Fig. 5-19 Card Detection Method 2

- Method3: Using card detection unit in GRF, outputting *sdmmc_detect_dual_edge_int*, 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 5-20 Register for SDMMC Card Detection Method 3

| Signal Name | Source | Default | Description |
|--------------------------------------|----------------------------|----------------|---|
| sd_detect_rising_edge_dectect_en | GRF_SIG_DETECT_CON [0] | 0 | sdmmc detect_n signal rise edge interrupt enable. 1'b1: enable 1'b0: disable |
| sd_detect_fall_edge_detect_en | GRF_SIG_DETECT_CON [1] | 0 | sdmmc detect_n signal fall edge interrupt enable. 1'b1: enable 1'b0: disable |
| sd_detect_filter_time_sel | GRF_SIG_DETECT_CON [15:14] | 0 | the time select for sd card detect filter: 2'b00: 5ms 2'b01: 15ms 2'b10: 35ms 2'b11: 50ms |
| sd_detect_rising_edge_dectect_clr | SIG_DETECT_CLR[0] | 0 | sdmmc detect_n signal rise edge interrupt clear. 1'b1: enable 1'b0: disable |
| sd_detect_fall_edge_detect_clr | SIG_DETECT_CLR[1] | 0 | sdmmc detect_n signal fall edge interrupt clear. 1'b1: enable 1'b0: disable |
| sd_detect_rising_edge_dectect_status | SIG_DETECT_STATUS[0] | 0 | sdmmc detect_n signal rise edge interrupt status. 1'b1: enable 1'b0: disable |
| sd_detect_fall_edge_detect_status | SIG_DETECT_STATUS[1] | 0 | sdmmc detect_n signal fall edge interrupt status. 1'b1: enable 1'b0: disable |

- Method4: Using card_detect_n for interrupt source, connecting to directly. Commonly for SDMMC/SDIO.

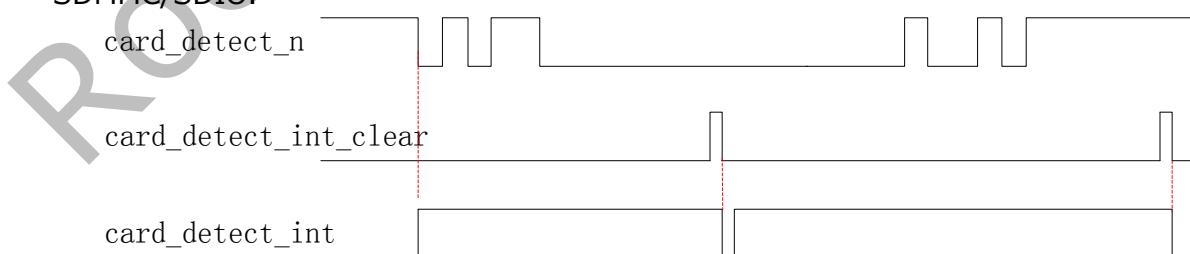


Fig. 5-20 Card Detection Method 4

Chapter 2 USB OTG 2.0

2.1 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.

2.1.1 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

2.2 Block Diagram

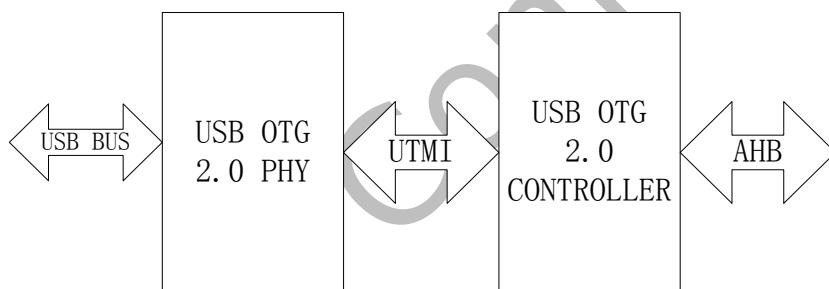


Fig. 5-1 USB OTG 2.0 Architecture

The Fig 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.

2.2.1 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.

2.2.2 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.

2.2.3 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.

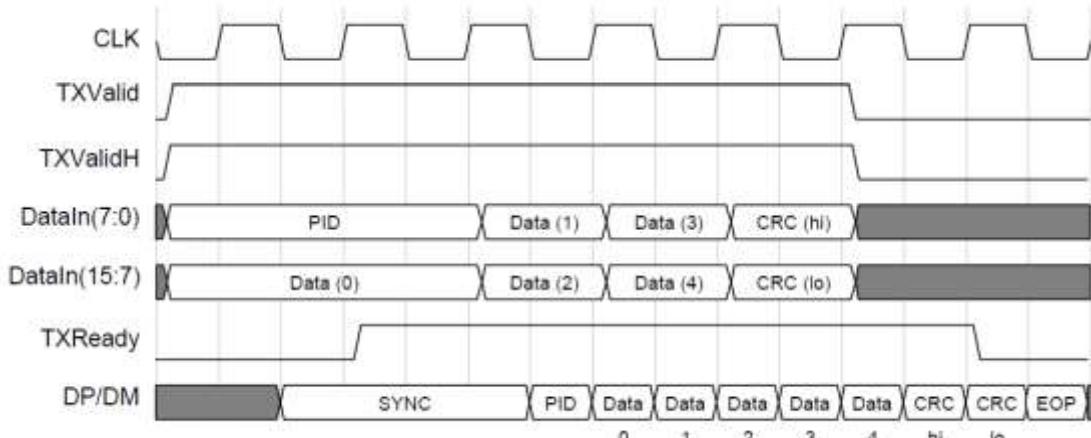


Fig. 5-2 UTMI interface – Transmit timing for a data packe

● 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.

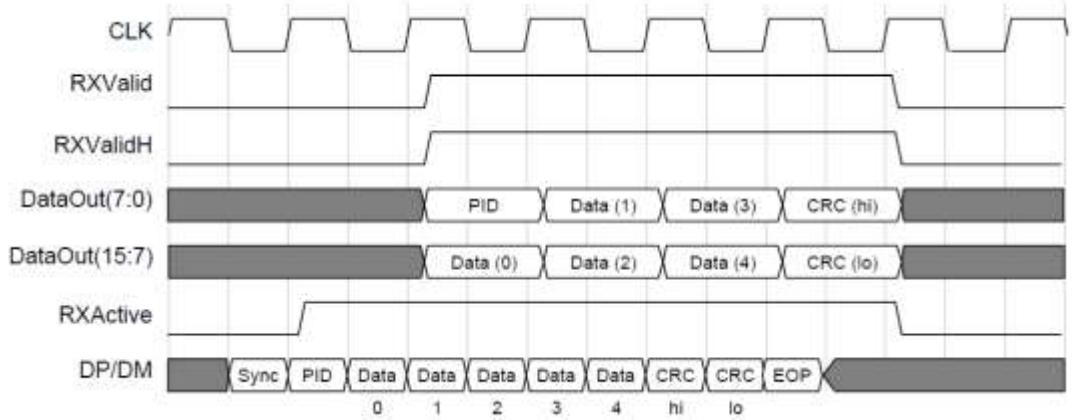


Fig. 5-3 UTMI interface – Receive timing for a data packet

2.3 USB OTG2.0 Controller

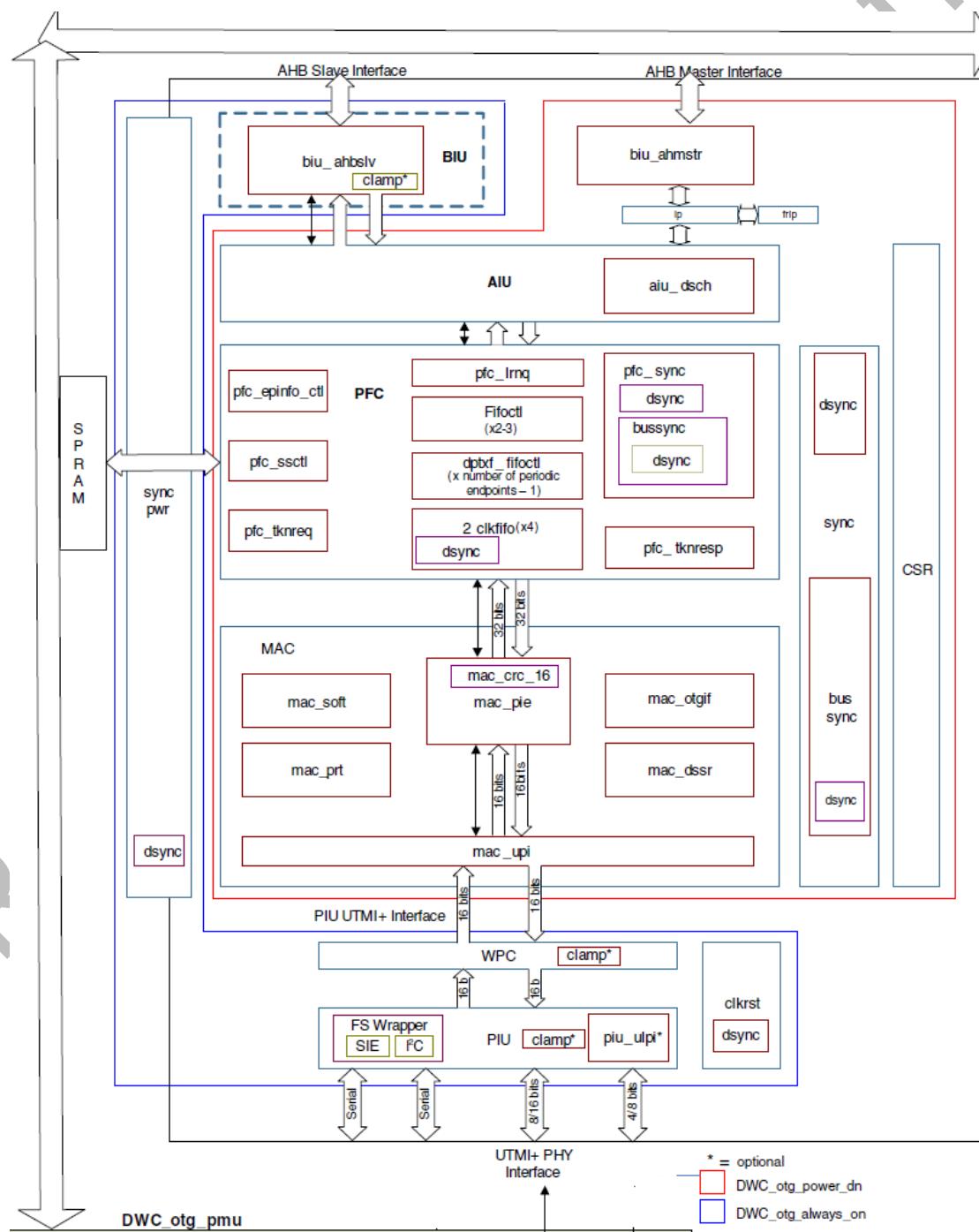


Fig. 5-4 USB OTG2.0 Controller Architecture

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.

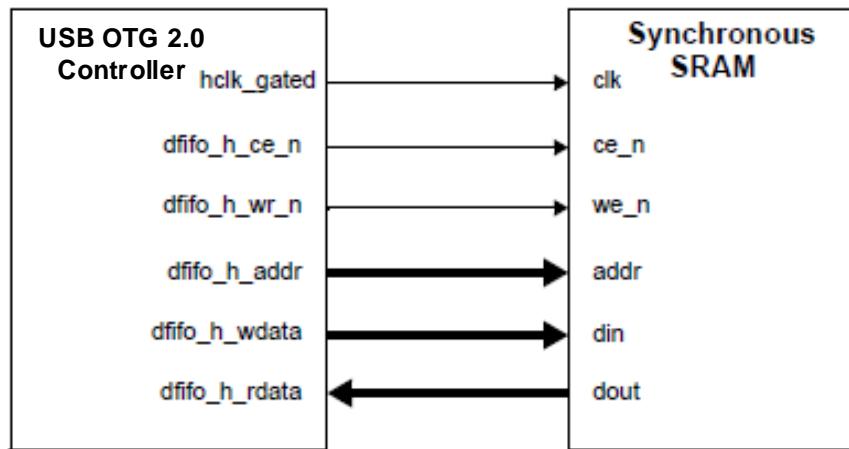


Fig. 5-5 DFIFO single-port synchronous SRAM interface

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 .

2.3.1 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.

2.3.2 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.

2.3.3 FIFO Mapping

- FIFO mapping in Host mode.

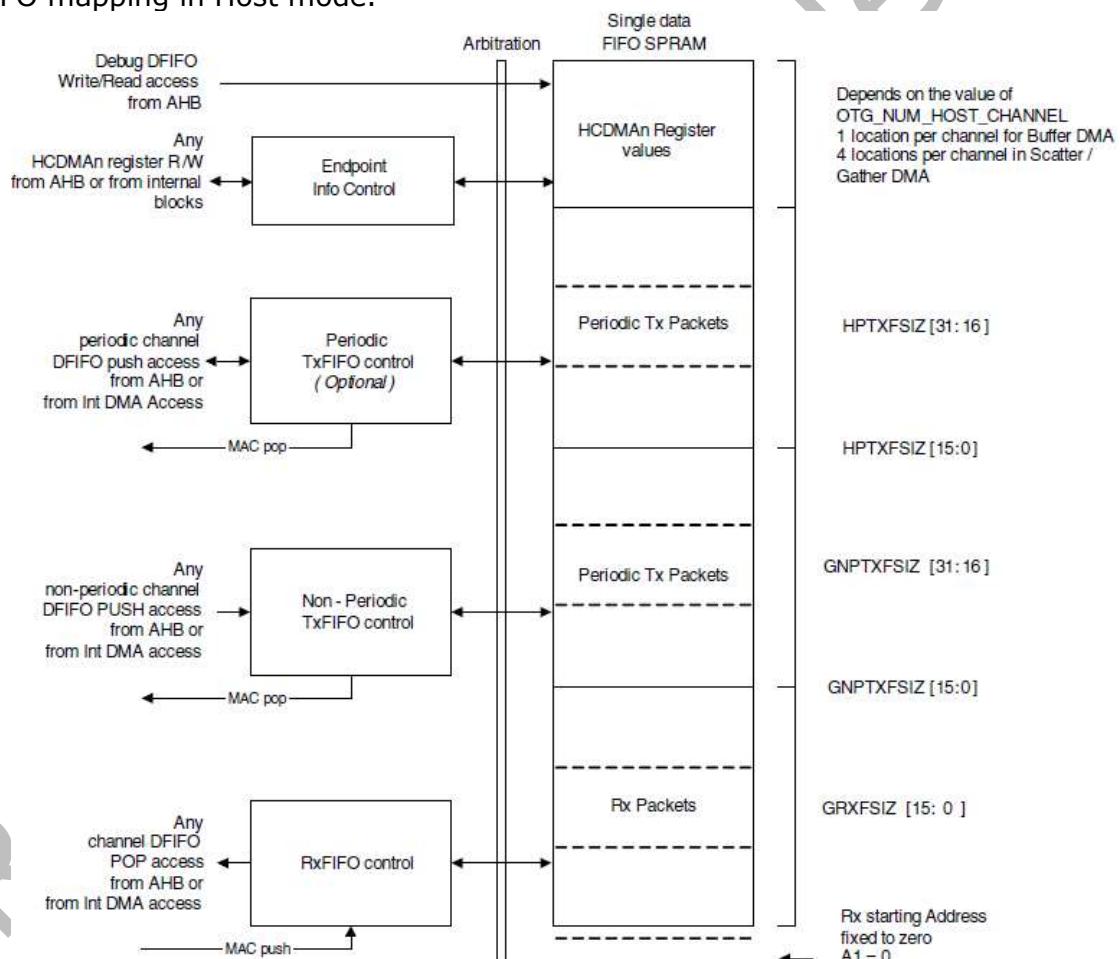


Fig 5-6 USB OTG 2.0 Controller host mode FIFO address mapping

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.

- 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 quad let information for each

endpoint direction.

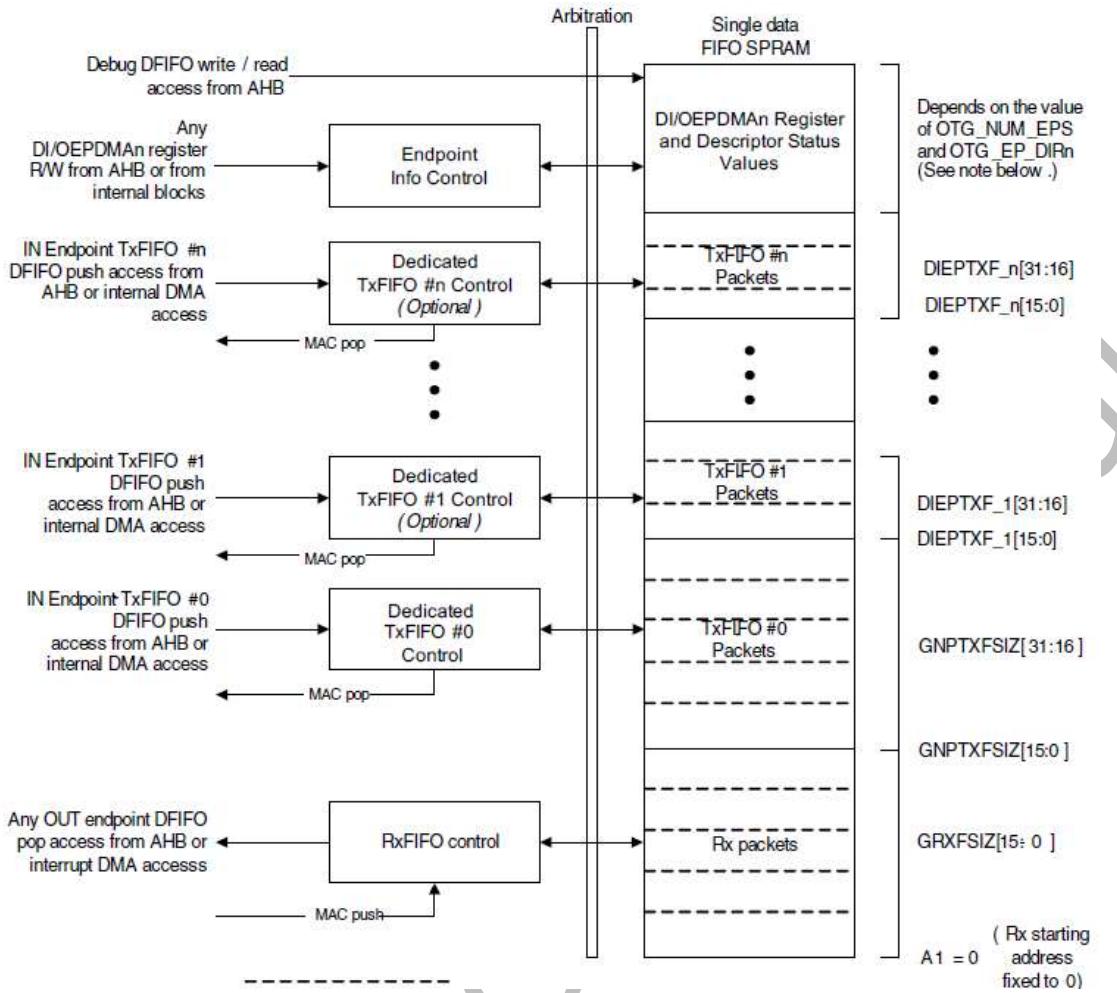


Fig 5-7 USB OTG 2.0 Controller device mode FIFO address mapping

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).

2.4 USB OTG2.0 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:

- provide dual UTMI ports
- OTG0 Support UART Bypass Function
- Fully compliant with USB specifications Rev 2.0, 1.1 HOST/Device and OTG 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
- 8 bit or 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
- Dual 3.3V / 1.2V supply

2.4.1 Block Diagram

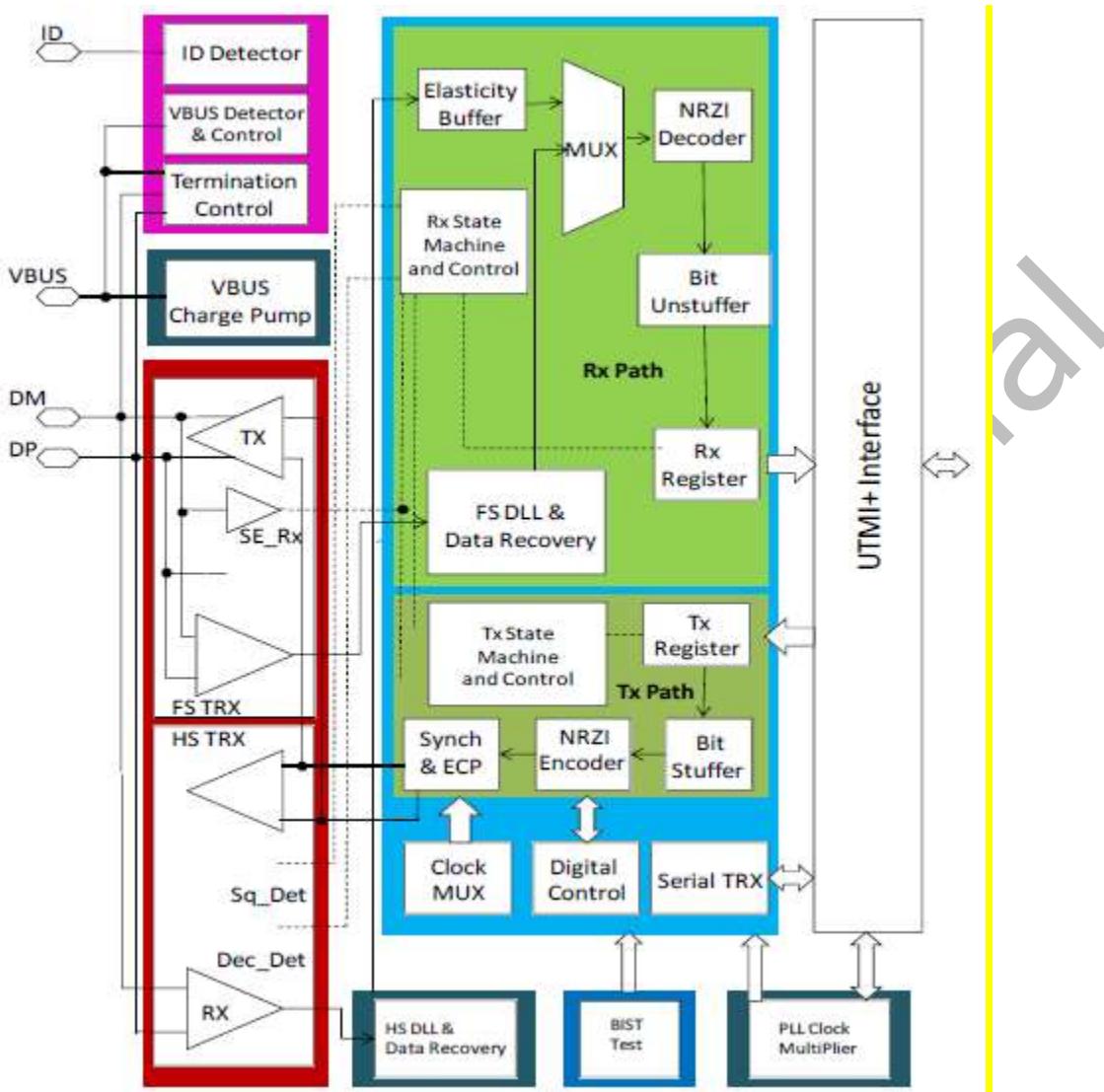


Fig. 5-8 usb phy architecture

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 XCSR 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 is 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 ± 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.

VBU S 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.

2.5 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 | grf_uoc1_con4[11] | I | DM0 Transmitter Digital Bypass Enable |
| BYPASSSEL0 | grf_uoc1_con4[13] | 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 | grf_uoc1_con4[10] | I | 1'b1: OTG0 disable; 1'b0: OTG0 normal mode |
| COMMONONNN | grf_uoc0_con5[11] | 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.

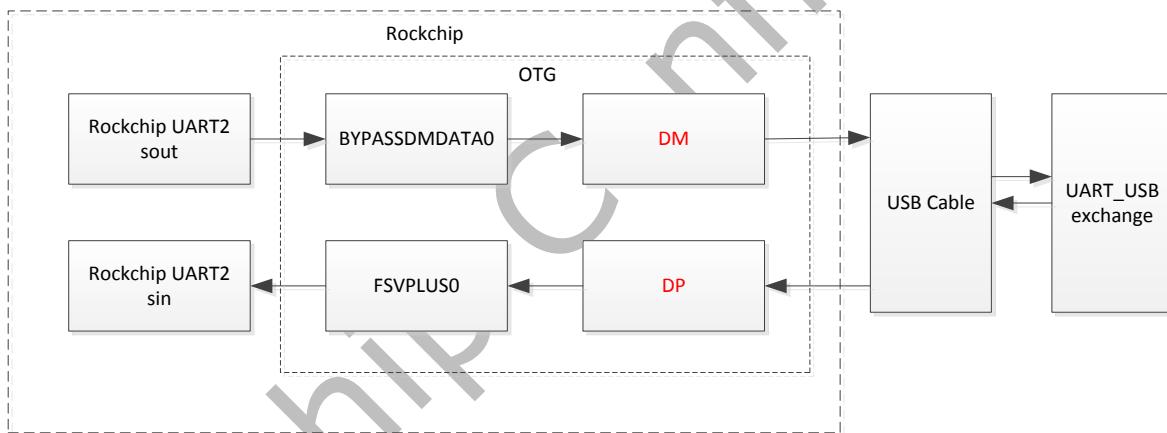


Fig 5-9 UART Application

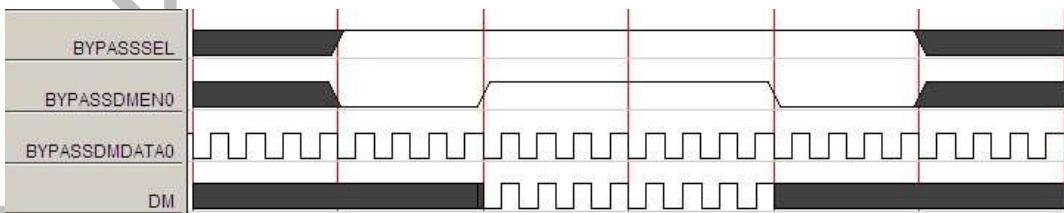


Fig. 5-10 UART Timing Sequence

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 DP 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 BYPASSDMENO, and BYPASSDMDATA0.
- To receive data, monitor FSVPLUS0.

To return to normal operating mode:

1. Ensure that there is no activity on the USB.
2. Set BYPASSSEL0 to 1'b0.
3. Set SUSPENDM0 to 1'b1. Resume the USB PHY.
4. Set COMMONONN to 1'b0.
5. set OTGDISABLE0 to 1'b0.

2.6 Register Description

2.6.1 Register Summary

| 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_GRXFISIZ | 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 Purpost 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_GDFIFO CFG | 0x005c | W | 0x00000000 | Global DFIFO Software Config Register |
| USBOTG_GADPCTL | 0x0060 | W | 0x00000000 | ADP Timer,Control and Status Register |
| USBOTG_HPTXFSIZ | 0x0100 | W | 0x00000000 | Host Periodic Transmit FIFO Size Register |
| USBOTG_DIEPTXFn | 0x0104 | W | 0x00000000 | Device Periodic Transmit FIFO-n Size Register |
| USBOTG_HCFG | 0x0400 | W | 0x00000000 | Host Configuration Register |

| Name | Offset | Size | Reset Value | Description |
|-------------------|--------|------|-------------|--|
| USBOTG_HFIR | 0x0404 | W | 0x00000000 | Host Frame Interval Register |
| USBOTG_HFNUM | 0x0408 | W | 0x0000ffff | Host Frame Number/Frame Time Remaining Register |
| USBOTG_HPTXSTS | 0x0410 | W | 0x00000000 | Host Periodic Transmit FIFO/Queue Status Register |
| USBOTG_HAINT | 0x0414 | W | 0x00000000 | Host All Channels Interrupt Reigster |
| 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 | W | 0x00000000 | Host Channel-n Characteristics Register |
| USBOTG_HCSPLTn | 0x0504 | W | 0x00000000 | Host Channel-n Split Control Register |
| USBOTG_HCINTn | 0x0508 | W | 0x00000000 | Host Channel-n Interrupt Register |
| USBOTG_HCINTMSKn | 0x050c | W | 0x00000000 | Host Channel-n Interrupt Mask Register |
| USBOTG_HCTSIZn | 0x0510 | W | 0x00000000 | Host Channel-n Transfer Size Register |
| USBOTG_HCDMAAn | 0x0514 | W | 0x00000000 | Host Channel-n DMA Address Register |
| USBOTG_HCDMABn | 0x051c | W | 0x00000000 | Host Channel-n DMA Buffer Address Register |
| USBOTG_DCFG | 0x0800 | W | 0x08200000 | Device Cconfiguration 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 |
| USBOTG_DTKNQR2 | 0x0824 | W | 0x00000000 | Device IN token sequence learning queue read register2 |
| USBOTG_DVBUUSDIS | 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 |

| Name | Offset | Size | Reset Value | Description |
|---------------------|--------|------|-------------|---|
| 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 | W | 0x00000000 | Device each IN endpoint -n interrupt Register |
| USBOTG_DOEPEACHMSKn | 0x0880 | W | 0x00000000 | Device each out endpoint-n interrupt register |
| USBOTG_DIEPCTL0 | 0x0900 | W | 0x00008000 | Device control IN endpoint 0 control register |
| USBOTG_DIEPINTn | 0x0908 | W | 0x00000000 | Device Endpoint-n Interrupt Register |
| USBOTG_DIEPTSIzn | 0x0910 | W | 0x00000000 | Device endpoint n transfer size register |
| USBOTG_DIEPDMAAn | 0x0914 | W | 0x00000000 | Device endpoint-n DMA address register |
| USBOTG_DTXFSTSn | 0x0918 | W | 0x00000000 | Device IN endpoint transmit FIFO status register |
| USBOTG_DIEPDMABn | 0x091c | W | 0x00000000 | Device endpoint-n DMA buffer address register |
| USBOTG_DIEPCTLn | 0x0920 | W | 0x00000000 | Device endpoint-n control register |
| USBOTG_DOEPCTL0 | 0x0b00 | W | 0x00000000 | Device control OUT endpoint 0 control register |
| USBOTG_DOEPINTn | 0x0b08 | W | 0x00000000 | Device endpoint-n control register |
| USBOTG_DOEPTSIzn | 0x0b10 | W | 0x00000000 | Device endpoint n transfer size register |
| USBOTG_DOEPDMAAn | 0x0b14 | W | 0x00000000 | Device Endpoint-n DMA Address Register |
| USBOTG_DOEPDMABn | 0x0b1c | W | 0x00000000 | Device endpoint-n DMA buffer address register |
| USBOTG_DOEPCTLn | 0x0b20 | W | 0x00000000 | Device endpoint-n control register |
| USBOTG_PCGCR | 0x0b24 | W | 0x200b8000 | Power and clock gating control register |
| USBOTG_EPBUFO | 0x1000 | W | 0x00000000 | Device endpoint 0 / host out channel 0 address |
| USBOTG_EPBUF1 | 0x2000 | W | 0x00000000 | Device endpoint 1 / host out channel 1 address |
| USBOTG_EPBUF2 | 0x3000 | W | 0x00000000 | Device endpoint 2 / host out channel 2 address |
| USBOTG_EPBUF3 | 0x4000 | W | 0x00000000 | Device endpoint 3 / host out channel 3 address |
| USBOTG_EPBUF4 | 0x5000 | W | 0x00000000 | Device endpoint 4 / host out channel 4 address |

| Name | Offset | Size | Reset Value | Description |
|---------------|--------|------|-------------|--|
| USBOTG_EPBUF5 | 0x6000 | W | 0x00000000 | Device endpoint 5 / host out channel 5 address |
| USBOTG_EPBUF6 | 0x7000 | W | 0x00000000 | Device endpoint 6 / host out channel 6 address |
| USBOTG_EPBUF7 | 0x8000 | W | 0x00000000 | Device endpoint 7 / host out channel 7 address |

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

2.6.2 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. |
| 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. 0: OTG Version 1.3. In this version the core supports Data line pulsing and VBus pulsing for SRP. 1: OTG Version 2.0. In this version the core supports only Data line pulsing for SRP. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 19 | RO | 0x0 | <p>BSesVld B-session valid Indicates the Device mode transceiver status. 0: B-session is not valid. 1: B-session is valid.</p> <p>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.</p> |
| 18 | RO | 0x0 | <p>ASesVld A-session valid Indicates the Host mode transceiver status. 0: A-session is not valid 1: A-session is valid</p> <p>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.</p> |
| 17 | RO | 0x0 | <p>DbnTime Long/short debounce time Indicates the debounce time of a detected connection. 0: Long debounce time, used for physical connections (100 ms + 2.5 us) 1: Short debounce time, used for soft connections (2.5 us)</p> |
| 16 | RO | 0x0 | <p>ConIDSts Connector ID Status Indicates the connector ID status on a connect event. 0: The core is in A-Device mode 1: The core is in B-Device mode</p> |
| 15:12 | RO | 0x0 | reserved |
| 11 | RW | 0x0 | <p>DevHNPEn Device HNP Enable The application sets this bit when it successfully receives a SetFeature. SetHNPEnable command from the connected USB host. 0: HNP is not enabled in the application 1: HNP is enabled in the application</p> |
| 10 | RW | 0x0 | <p>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. 0: Host Set HNP is not enabled 1: Host Set HNP is enabled</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 9 | RW | 0x0 | <p>HNPReq HNP request</p> <p>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.</p> <p>0: No HNP request 1: HNP request</p> |
| 8 | RO | 0x0 | <p>HstNegScs Host Negotiation Success</p> <p>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.</p> <p>0: Host negotiation failure 1: Host negotiation success</p> |
| 7:2 | RO | 0x0 | reserved |
| 1 | RW | 0x0 | <p>SesReq Session Request</p> <p>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.</p> <p>0: No session request 1: Session request</p> |
| 0 | RO | 0x0 | <p>SesReqScs Session Request Success</p> <p>The core sets this bit when a session request initiation is successful.</p> <p>0: Session request failure 1: Session request success</p> |

USBOTG_GOTGINT

Address: Operational Base + offset (0x0004)
 Interrupt Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|-------------|
| 31:21 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 connect. 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 |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:23 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 22 | RW | 0x0 | <p>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.</p> <p>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.</p> <p>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.</p> |
| 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. 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>0: GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is half empty</p> <p>1: GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is completely empty</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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</p> <p>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</p> <p>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 0: Core operates in Slave mode 1: Core operates in a DMA mode This bit is always 0 when Slave-Only mode has been selected.</p> |
| 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> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 0 | RW | 0x0 | <p>GlblIntrMsk Global Interrupt Mask</p> <p>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

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 31 | RW | 0x0 | <p>CorruptTxpacket Corrupt Tx packet</p> <p>This bit is for debug purposes only. Never set this bit to 1.</p> |
| 30 | RW | 0x0 | <p>ForceDevMode Force Device Mode</p> <p>Writing a 1 to this bit forces the core to device mode irrespective of utmiotg_iddig input pin.</p> <p>1'b0: Normal Mode 1'b1: Force Device Mode</p> <p>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</p> <p>Writing a 1 to this bit forces the core to host mode irrespective of utmiotg_iddig input pin.</p> <p>1'b0: Normal Mode 1'b1: Force Host Mode</p> <p>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</p> <p>Writing a 1 to this bit enables the TxEndDelay timers in the core.</p> <p>1'b0: Normal mode 1'b1: Introduce Tx end delay timers</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 27 | RW | 0x0 | <p>IC_USBTrfCtl IC_USB TrafficPullRemove 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> |
| 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. 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. 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. 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 1'b0: PHY does not invert ExternalVbusIndicator signal 1'b1: PHY does invert ExternalVbusIndicator signal</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 22 | RW | 0x0 | <p>TermSelDLPulse TermSel DLine Pulsing Selection This bit selects utmi_termselect to drive data line pulse during SRP. 1'b0: Data line pulsing using utmi_txvalid (default). 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. 1'b0: PHY uses internal VBUS valid comparator. 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. 1'b0: PHY drives VBUS using internal charge pump (default). 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. 1'b0: PHY powers down internal clock during suspend. 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. 1'b0: PHY does not use AutoResume feature. 1'b1: PHY uses AutoResume feature. (Valid only when RTL parameter OTG_HSPHY_INTERFACE = 2 or 3)</p> |
| 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. 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> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 16 | RW | 0x0 | <p>OtgI2CSel UTMIFS or I2C Interface Select The application uses this bit to select the I2C interface. 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. 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_fs_ls_low_power core output signal, and is valid only for UTMI+ PHYs.</p> |
| 14 | RO | 0x0 | reserved |
| 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 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 DWC_otg core's HNP capabilities. 0: HNP capability is not enabled. 1: 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> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 8 | RW | 0x0 | <p>SRPCap SRP-Capable</p> <p>The application uses this bit to control the DWC_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.</p> <p>This bit is writable only if an SRP mode was specified for Mode of Operation in coreConsultant (parameter OTG_MODE). Otherwise, reads return 0.</p> |
| 7 | RW | 0x0 | <p>DDRSel ULPI DDR Select</p> <p>The application uses this bit to select a Single Data Rate (SDR) or Double Data Rate (DDR) or ULPI interface.</p> <p>0: Single Data Rate ULPI Interface, with 8-bit-wide data bus 1: Double Data Rate ULPI Interface, with 4-bit-wide data bus</p> |
| 6 | RW | 0x0 | <p>PHYSel</p> <p>USB 2.0 High-Speed PHY or USB 1.1 Full-Speed Serial Transceiver</p> <p>The application uses this bit to select either a high-speed UTMI+ or ULPI PHY, or a full-speed transceiver.</p> <p>0: USB 2.0 high-speed UTMI+ or ULPI PHY 1: USB 1.1 full-speed serial transceiver</p> <p>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.</p> <p>If both interface types were selected in coreConsultant (parameters have non-zero values), the application uses this bit to select which interface is active, and access is Read and Write.</p> |
| 5 | RW | 0x0 | <p>FSIntf Full-Speed Serial Interface Select</p> <p>The application uses this bit to select either a unidirectional or bidirectional USB 1.1 full-speed serial transceiver interface.</p> <p>0: 6-pin unidirectional full-speed serial interface 1: 3-pin bidirectional full-speed serial interface</p> <p>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> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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. 0: UTMI+ Interface 1: ULPI Interface This bit is writable only if UTMI+ and ULPI was specified for High-Speed PHY Interface(s) in coreConsultant configuration (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. 0: 8 bits 1: 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> |
| 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 interpacket 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. 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: High-speed operation: One 30-MHz PHY clock = 16 bit times One 60-MHz PHY clock = 8 bit times 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 |
| 10:6 | RW | 0x00 | TxFNum TxFIFO Number This is the FIFO number that must be flushed using the TxFIFO Flush bit. This field must not be changed until the core clears the TxFIFO Flush bit. 5'h0: Non-periodic TxFIFO flush in Host mode; Non-periodic TxFIFO flush in device mode when in shared FIFO operation. Tx FIFO 0 flush in device mode when in dedicated FIFO mode. 5'h1: Periodic TxFIFO flush in Host mode: Periodic TxFIFO 1 flush in Device mode when in shared FIFO operation; TXFIFO 1 flush in device mode when in dedicated FIFO mode. 5'h2: Periodic TxFIFO 2 flush in Device mode when in shared FIFO operation: TXFIFO 2 flush in device mode when in dedicated FIFO mode. ... 5'hF: Periodic TxFIFO 15 flush in Device mode when in shared FIFO operation: TXFIFO 15 flush in device mode when in dedicated FIFO mode. 5'h10: Flush all the transmit FIFOs in device or host mode. |
| 5 | R/W SC | 0x0 | TxFFlsh TxFIFO Flush 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 TxFIFO nor reading from the TxFIFO. 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. 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 4 | R/W SC | 0x0 | RxFFlsh Rx FIFO Flush 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. |
| 3 | R/W SC | 0x0 | INTknQFlsh IN Token Sequence Learning Queue Flush This bit is valid only if OTG_ENDED_TX_FIFO = 0. The application writes this bit to flush the IN Token Sequence Learning Queue. |
| 2 | W1 C | 0x0 | FrmCntrRst Host Frame Counter Reset 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. |
| 1 | R/W SC | 0x0 | Reset A write to this bit issues a soft reset to the DWC_otg_power_dn module of the core. |

| Bit | Attr | Reset Value | Description |
|-----|-----------|-------------|--|
| 0 | R/W SC | 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"> CGCCTL.RstPdwnModule PCGCCTL.GateHclk PCGCCTL.PwrClmp PCGCCTL.StopPPhyLPwrClkSelClk GUSBCFG.PhyLPwrClkSel GUSBCFG.DDRSel GUSBCFG.PHYSel GUSBCFG.FSIntf GUSBCFG.ULPI_UTMI_Sel GUSBCFG.PHYIf HCFG.FSLSPclkSel DCFG.DevSpd GGPIO GPWRDN GADPCTL <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 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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31 | W1C | 0x0 | <p>WkUpInt Resume/Remote Wakeup Detected Interrupt Wakeup Interrupt during Suspend(L2) or LPM(L1) state. 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. 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 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>DisconnInt Disconnect Detected Interrupt This interrupt is asserted when a device disconnect is detected.</p> |
| 28 | W1C | 0x0 | <p>ConIDStsChng Connector ID Status Change This interrupt is asserted when there is a change in connector ID status.</p> |
| 27 | W1C | 0x0 | <p>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.</p> |
| 26 | RO | 0x0 | <p>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).</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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 DWC_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. |
| 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. For example, after detecting an endpoint mismatch, the application: Sets a global non-periodic IN NAK handshake, Disables In endpoints, Flushes the FIFO, Determines the token sequence from the IN Token Sequence Learning Queue, Re-enables the endpoints, Clears the global non-periodic IN NAK handshake. If the global non-periodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an "IN token received when FIFO empty" interrupt. The OTG then sends the host a NAK response. To avoid this scenario, the application can check the GINTSTS.FetSusp interrupt, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the IN token received when FIFO empty interrupt when clearing a global IN NAK handshake. |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 21 | W1C | 0x0 | <p>incomlP Incomplete Periodic Transfer</p> <p>In Host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending which are scheduled for the current microframe. 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 microframe. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register.</p> |
| 20 | W1C | 0x0 | <p>incompISOIN Incomplete Isochronous IN Transfer</p> <p>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 microframe. 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.</p> |
| 19 | RO | 0x0 | <p>OEPInt OUT Endpoints Interrupt</p> <p>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.</p> |
| 18 | RO | 0x0 | <p>IEPInt IN Endpoints Interrupt</p> <p>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.</p> |
| 17 | W1C | 0x0 | <p>EPMIs Endpoint Mismatch Interrupt</p> <p>Note: This interrupt is valid only in shared FIFO operation.</p> <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 16 | W1 C | 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 | W1 C | 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 | W1 C | 0x0 | ISOOutDrop Isochronous OUT Packet Dropped Interrupt The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum packet size packet for the isochronous OUT endpoint. |
| 13 | W1 C | 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 | W1 C | 0x0 | USBRst USB Reset The core sets this bit to indicate that a reset is detected on the USB. |
| 11 | W1 C | 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 | W1 C | 0x0 | ErlySusp Early Suspend The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms. |
| 9 | W1 C | 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 8 | W1C | 0x0 | <p>ULPICKINT ULPI Carkit Interrupt</p> <p>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)</p> <p>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.</p> |
| 7 | RO | 0x0 | <p>GOUTNakEff Global OUT NAK Effective</p> <p>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).</p> |
| 6 | RO | 0x0 | <p>GINNakEff Global IN Non-Periodic NAK Effective</p> <p>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.</p> |
| 5 | RO | 0x0 | <p>NPTxFEmp Non-Periodic TxFIFO Empty</p> <p>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).</p> |
| 4 | RO | 0x0 | <p>RxFLvl RxFIFO Non-Empty</p> <p>Indicates that there is at least one packet pending to be read from the RxFIFO.</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 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. |
| 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

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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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 |
| 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. |

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:16 | RW | 0x0000 | <p>NPTxFDep Non-periodic TxFIFO 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.</p> |
| 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): 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 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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]: 2'b00: IN/OUT token 2'b01: Zero-length transmit packet (device IN/host OUT) 2'b10: PING/CSPLIT token 2'b11: Channel halt command Bit [24]: Terminate (last entry for selected channel/endpoint)</p> |
| 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 <=n <= 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 <=n <=32,768) 16'h8000: 32,768 words available Others: Reserved</p> |

USBOTG_GI2CCTL

Address: Operational Base + offset (0x0030)

I2C Address Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31 | R/W SC | 0x0 | <p>BsyDne I2C Busy/Done</p> <p>The application sets this bit to 1'b1 to start a request on the I2C interface. When the transfer is complete, the core deasserts 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> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 |
| 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. |

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

| Bit | Attr | Reset Value | Description |
|-------|--------|-------------|--|
| 31 | R/W SC | 0x0 | DisUlpiDrv Disable ULPI Drivers This field is used only if the Carkit interface was enabled in coreConsultant (parameter 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 DWC_otg core disables drivers for output signals and masks input signal for the ULPI interface. DWC_otg clears this bit before enabling the ULPI interface. |
| 30:28 | RO | 0x0 | reserved |
| 27 | R/W SC | 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). |
| 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/W SC | 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. |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RW | 0x00000000 | UserID Application-programmable ID field. |

USBOTG_GSNPSID

Address: Operational Base + offset (0x0040)

Core ID Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RO | 0x00004f54 | CoreID Release number of the core being used |

USBOTG_GHWCFG1

Address: Operational Base + offset (0x0044)

User HW Config1 Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RO | 0x00000000 | <p>epdir Endpoint Direction This 32-bit field uses two bits per endpoint to determine the endpoint direction.</p> <p>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)</p> <p>Direction 2'b00: BIDIR (IN and OUT) endpoint 2'b01: IN endpoint 2'b10: OUT endpoint 2'b11: Reserved</p> |

USBOTG_GHWCFG2

Address: Operational Base + offset (0x0048)

User HW Config2 Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31 | RO | 0x0 | <p>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.</p> |
| 30:26 | RO | 0x00 | <p>TknQDepth Device Mode IN Token Sequence Learning Queue Depth Range: 0-30</p> |
| 25:24 | RO | 0x0 | <p>PTxQDepth Host Mode Periodic Request Queue Depth 2'b00: 2 2'b01: 4 2'b10: 8 Others: Reserved</p> |
| 23:22 | RO | 0x0 | <p>NPTxQDepth Non-periodic Request Queue Depth 2'b00: 2 2'b01: 4 2'b10: 8 Others: Reserved</p> |
| 21 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. |
| 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 |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 2:0 | RO | 0x0 | <p>OtgMode Mode of Operation</p> <p>3'b000: HNP- and SRP-Capable OTG (Host and Device)</p> <p>3'b001: SRP-Capable OTG (Host and Device)</p> <p>3'b010: Non-HNP and Non-SRP Capable OTG (Host and Device)</p> <p>3'b011: SRP-Capable Device</p> <p>3'b100: Non-OTG Device</p> <p>3'b101: SRP-Capable Host</p> <p>3'b110: Non-OTG Host</p> <p>Others: Reserved</p> |

USBOTG_GHWCFG3

Address: Operational Base + offset (0x004c)

User HW Config3 Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 31:16 | RO | 0x0000 | <p>DfifoDepth DFIFO Depth</p> <p>This value is in terms of 32-bit words.</p> <p>Minimum value is 32</p> <p>Maximum value is 32,768</p> |
| 15 | RO | 0x0 | <p>OTG_ENABLE_LPM</p> <p>LPM mode specified for Mode of Operation (parameter OTG_ENABLE_LPM).</p> |
| 14 | RO | 0x0 | <p>OTG_BC_SUPPORT</p> <p>This bit indicates the HS OTG controller support for Battery Charger.</p> <p>0 : No Battery Charger Support</p> <p>1 : Battery Charger support present.</p> |
| 13 | RO | 0x0 | <p>OTG_ENABLE_HSIC</p> <p>HSIC mode specified for Mode of Operation (parameter OTG_ENABLE_HSIC).</p> <p>Value Range: 0-1</p> <p>1: HSIC-capable with shared UTMI PHY interface</p> <p>0: Non-HSIC-capable</p> |
| 12 | RO | 0x0 | <p>OTG_AD_P_SUPPORT</p> <p>This bit indicates whether ADP logic is present within or external to the HS OTG controller</p> <p>0: No ADP logic present with HSOTG controller</p> <p>1: ADP logic is present along with HSOTG controller.</p> |
| 11 | RO | 0x0 | <p>RstType</p> <p>Reset Style for Clocked always Blocks in RTL</p> <p>1'b0: Asynchronous reset is used in the core</p> <p>1'b1: Synchronous reset is used in the core</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 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 |
| 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 DWC_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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 |
| 29:26 | RO | 0x0 | INEndps 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 | NumCtlEndps Number of Device Mode Control Endpoints in Addition to Endpoint Range: 0-15 |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 15:14 | RO | 0x0 | <p>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</p> |
| 13:7 | RO | 0x0 | reserved |
| 6 | RO | 0x0 | <p>EnHiber Enable Hibernation 1'b0: Hibernation feature not enabled 1'b1: Hibernation feature enabled</p> |
| 5 | RO | 0x0 | <p>AhbFreq Minimum AHB Frequency Less Than 60 MHz 1'b0: No 1'b1: Yes</p> |
| 4 | RO | 0x0 | <p>EnParPwrDown Enable Partial Power Down 1'b0: Partial Power Down Not Enabled 1'b1: Partial Power Down Enabled</p> |
| 3:0 | RO | 0x0 | <p>NumDevPerioEps Number of Device Mode Periodic IN Endpoints Range: 0-15</p> |

USBOTG_GLPMCFG

Address: Operational Base + offset (0x0054)

Core LPM Configuration Register

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

| Bit | Attr | Reset Value | Description |
|-------|--------|-------------|---|
| 31 | RW | 0x0 | <p>InvSelHsic HSIC-Invert Select HSIC</p> <p>The application uses this bit to control the DWC_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 HSICcapable PHYs. If the if_sel_hsic input signal is1:</p> <p>1'b1: HSIC capability is not enabled.</p> <p>1'b0: HSIC capability is enabled,</p> <p>If InvSelHsic = 1'b0: HSIC capability is enabled. If the if_sel_hsic input signal is 0:</p> <p>1'b1: HSIC capability is enabled,</p> <p>1'b0: HSIC capability is not enabled.</p> <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 & 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> |
| 24 | RW | 0x0 | <p>SndLPM Send LPM Transaction</p> <p>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.</p> |
| 23:21 | R/W SC | 0x0 | <p>LPM_Retry_Cnt LPM Retry Count</p> <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 20:17 | RW | 0x0 | <p>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.</p> |
| 16 | RO | 0x0 | <p>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</p> |
| 15 | RO | 0x0 | <p>SlpSts Port Sleep Status 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. The core comes out of sleep: When there is any activity on the USB line_state 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. Host Mode: The host transitions to the Sleep (L1) state as a side effect 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). Values: 1'b0: Core not in L1 1'b1: Core in L1</p> |

| Bit | Attr | Reset Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|------------|-----------------|---------------------------|--|--------|-----------------|---------------------------|---|---------|----|---|---------|-----|---|---------|-----|---|---------|-----|---|---------|-----|---|---------|-----|---|---------|-----|---|---------|-----|---|---------|-----|----|---------|-----|----|---------|-----|----|---------|-----|----|---------|-----|----|---------|---------|----|---------|---------|----|---------|---------|
| 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>11: ACK 10: NYET 01: STALL 00: 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 TL1HubDrvResume2.</p> <table> <thead> <tr> <th>SI. No</th> <th>HIRD_Thres[3:0]</th> <th>Host mode 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> | SI. No | HIRD_Thres[3:0] | Host mode 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 |
| SI. No | HIRD_Thres[3:0] | Host mode 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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. 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. When a ULPI interface is configured, this bit must always be set if you are using the ULPI PHY. Note: For ULPI interfaces, 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. 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> |

| Bit | Attr | Reset Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------|-----------|-------------|---|--------|-----------|------------|---|---------|----|---|---------|-----|---|---------|-----|---|---------|-----|---|---------|-----|---|---------|-----|---|---------|-----|---|---------|-----|---|---------|-----|----|---------|-----|----|---------|-----|----|---------|-----|----|---------|-----|----|---------|------|----|---------|------|----|---------|------|
| 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 TL1HubDrvResume1 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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: 1: 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) 0: 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> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 0 | RW | 0x0 | <p>LPMCap LPM-Capable</p> <p>The application uses this bit to control the DWC_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.</p> <p>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_GPWRDN

Address: Operational Base + offset (0x0058)

Global Power Down Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 31:29 | RO | 0x0 | reserved |
| 28:24 | RO | 0x00 | <p>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</p> <p>These bits are present only if OTG_BC_SUPPORT = 1. Otherwise, these bits are reserved and will read 5'h0.</p> |
| 23 | W1C | 0x0 | <p>ADPInt ADP Interupt This bit is set whenever there is a ADP event.</p> |
| 22 | RO | 0x0 | <p>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.</p> |
| 21 | RO | 0x0 | <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 20:19 | RO | 0x0 | <p>LineState This field indicates the current linestate on USB as seen by the PMU module.</p> <p>2'b00: DM = 0, DP = 0. 2'b01: DM = 0, DP = 1. 2'b10: DM = 1, DP = 0. 2'b11: Not-defined.</p> <p>This bit is valid only when GPWRDN.PMUActv is 1.</p> |
| 18 | RW | 0x0 | <p>StsChngIntMsk Mask For StsChng Interrupt</p> |
| 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_PWRLOPT = 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_PWRLOPT = 2.</p> |
| 12 | RW | 0x0 | <p>DisconnectDetectMsk Mask For DisconnectDetect Interrupt</p> <p>This bit is valid only when OTG_EN_PWRLOPT = 2.</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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> |
| 5 | RW | 0x0 | <p>PwrDnSwtch</p> <p>Power Down Switch</p> <p>This bit indicates to the DWC_otg core VDD switch is in ON/OFF state</p> <p>1'b0: DWC_otg is in ON state 1'b1: DWC_otg is in OFF state</p> <p><i>Note: This bit must not be written to during normal mode of operation.</i></p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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).</p> <p>1'b1: DWC_otg is in normal operation 1'b0: reset DWC_otg</p> <p><i>Note: This bit must not be written to during normal mode of operation.</i></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.</p> <p>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.</p> <p>1'b0: DWC_otg in normal mode of operation 1'b1: DWC_otg in restore mode</p> <p><i>Note: This bit must not be written to during normal mode of operation. This bit is valid only when OTG_EN_PWR_OPT = 2.</i></p> |
| 1 | RW | 0x0 | <p>PMUActv PMU Active This is bit is to enable or disable the PMU logic.</p> <p>1'b0: Disable PMU module 1'b1: Enable PMU module</p> <p><i>Note: This bit must not be written to during normal mode of operation.</i></p> |
| 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 DWC_otg_core interrupt is selected 1'b1: the external DWC_otg_pmu interrupt is selected</p> <p><i>Note: This bit must not be written to during normal mode of operation.</i></p> |

USBOTG_GDFIFOCFG

Address: Operational Base + offset (0x005c)

Global DFIFO Software Config Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:16 | RW | 0x0000 | EPIInfoBaseAddr This field provides the start address of the EP info controller. |
| 15:0 | RW | 0x0000 | GDFIFOCfg 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. |

USBOTG_GADPCTL

Address: Operational Base + offset (0x0060)

ADP Timer,Control and Status Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:29 | RO | 0x0 | reserved |
| 28:27 | R/W SC | 0x0 | AR Access Request 2'b00 Read/Write Valid (updated by the core) 2'b01 Read 2'b10 Write 2'b11 Reserved |
| 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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]). |
| 19 | R/W SC | 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]). |

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 16:6 | RO | 0x000 | <p>RTIM RAMP TIME</p> <p>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:</p> <ul style="list-style-type: none"> 0x000 - 1 cycles 0x001 - 2 cycles 0x002 - 3 cycles and so on till 0x7FF - 2048 cycles <p>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.)</p> |
| 5:4 | RW | 0x0 | <p>PrbPer Probe Period</p> <p>These bits sets the TadpPrd as follows:</p> <ul style="list-style-type: none"> 2'b00 - 0.625 to 0.925 sec (typical 0.775 sec) 2'b01 - 1.25 to 1.85 sec (typical 1.55 sec) 2'b10 - 1.9 to 2.6 sec (typical 2.275 sec) 2'b11 - Reserved <p>(PRB_PER is also scaledown prb_per== 2'b00 => 400 ADP clocks prb_per== 2'b01 => 600 ADP clocks prb_per== 2'b10 => 800 ADP clocks prb_per==2'b11 => 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"> 2'b00 - 1 cycles 2'b01 - 2 cycles 2'b10 - 3 cycles 2'b11 - 4 cycles <p>For example if this value is chosen to 2'b01, it means that RTIM increments for every three 32Khz clock cycles.</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 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"> 2'b00 4 msec (Scaledown 2 32Khz clock cycles) 2'b01 8 msec (Scaledown 4 32Khz clock cycles) 2'b10 16 msec (Scaledown 8 32Khz clock cycles) 2'b11 32 msec (Scaledown 16 32Khz clock cycles) |

USBOTG_HPTXFSIZ

Address: Operational Base + offset (0x0100)

Host Periodic Transmit FIFO Size Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 31:16 | RW | 0x0000 | <p>PTxFSize Host Periodic TxFIFO Depth</p> <p>This value is in terms of 32-bit words.</p> <p>Minimum value is 16</p> <p>Maximum value is 32,768</p> <p>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</p> <p>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:</p> <p>In shared FIFO operation: OTG_RX_DFIFO_DEPTH + OTG_TX_NPERIO_DFIFO_DEPTH.</p> <p>In dedicated FIFO mode: OTG_RX_DFIFO_DEPTH + OTG_TX_HNPERIO_DFIFO_DEPTH.</p> <p>If Enable Dynamic FIFO Sizing? was deselected (parameter OTG_DFIFO_DYNAMIC = 0), these flops are optimized, and reads return the power-on value.</p> <p>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)

Device Periodic Transmit FIFO-n Size Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:16 | RW | 0x0000 | <p>INEP1TxDep IN Endpoint TxFIFO Depth This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768</p> <p>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 < n <= 15). If Enable Dynamic FIFO Sizing? was deselected (parameter OTG_DFIFO_DYNAMIC = 0), these flops are optimized, and reads return the power-on value.</p> <p>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> |
| 15:0 | RW | 0x0000 | <p>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)</p> <p>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 RxFIFO depth, you can write that value in this field. Programmed values must not exceed the power-on value set .</p> |

USBOTG_HCFG

Address: Operational Base + offset (0x0400)

Host Configuration Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:27 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 26 | RW | 0x0 | <p>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.</p> |
| 25:24 | RW | 0x0 | <p>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.</p> |
| 23 | RW | 0x0 | <p>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.</p> |
| 22:16 | RO | 0x0 | reserved |
| 15:8 | RW | 0x00 | <p>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.</p> |
| 7 | RW | 0x0 | <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 6:3 | RO | 0x0 | reserved |
| 2 | RW | 0x0 | <p>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.</p> <p>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</p> |
| 1:0 | RW | 0x0 | <p>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</p> |

USBOTG_HFIR

Address: Operational Base + offset (0x0404)

Host Frame Interval Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 31:16 | RO | 0x0 | reserved |
| 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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:16 | RO | 0x0000 | FrRem Frame Time Remaining Indicates the amount of time remaining in the current microframe (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. |
| 15:0 | RO | 0xfffff | 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. |

USBOTG_HPTXSTS

Address: Operational Base + offset (0x0410)
 Host Periodic Transmit FIFO/Queue Status Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:24 | RO | 0x00 | 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. 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) |
| 23:16 | RO | 0x00 | 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. 8'h0: Periodic Transmit Request Queue is full 8'h1: 1 location available 8'h2: 2 locations available n: n locations available (0 <=n <= 16) Others: Reserved |

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 15:0 | RW | 0x0000 | 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 16'h0: Periodic 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_HINT

Address: Operational Base + offset (0x0414)

Host All Channels Interrupt Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 31:16 | RO | 0x0 | reserved |
| 15:0 | RO | 0x0000 | HINT Channel Interrupts One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15 |

USBOTG_HINTMSK

Address: Operational Base + offset (0x0418)

Host All Channels Interrupt Mask Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 31:16 | RO | 0x0 | reserved |
| 15:0 | RW | 0x0000 | HAINTMsk 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

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

| Bit | Attr | Reset Value | Description |
|-------|--------|-------------|---|
| 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 | R/W SC | 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 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> |

| Bit | Attr | Reset Value | Description |
|-----|-----------|-------------|--|
| 7 | R/W SC | 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 suspend 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> |

| Bit | Attr | Reset Value | Description |
|-----|-----------|-------------|--|
| 6 | R/W SC | 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> |
| 4 | RO | 0x0 | <p>PrtOvrCurrAct Port Overcurrent Active Indicates the overcurrent condition of the port.</p> <p>1'b0: No overcurrent condition 1'b1: Overcurrent condition</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 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 0: No device is attached to the port. 1: A device is attached to the port. |

USBOTG_HCCHARn

Address: Operational Base + offset (0x0500)

Host Channel-n Characteristics Register

| Bit | Attr | Reset Value | Description |
|-----|--------|-------------|---|
| 31 | R/W SC | 0x0 | 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 |

| Bit | Attr | Reset Value | Description |
|-------|--------|-------------|---|
| 30 | R/W SC | 0x0 | <p>ChDis Channel Disable</p> <p>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</p> <p>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.</p> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame</p> <p>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</p> <p>This field selects the specific device serving as the data source or sink.</p> |
| 21:20 | RW | 0x0 | <p>MC_EC Multi Count (MC) / Error Count (EC)</p> <p>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 microframe 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 microframe 2'b11: 3 transactions to be issued for this endpoint per microframe</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> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 19:18 | RW | 0x0 | EPType Endpoint Type Indicates the transfer type selected. 2'b00: Control 2'b01: Isochronous 2'b10: Bulk 2'b11: Interrupt |
| 17 | RW | 0x0 | LSpdDev Low-Speed Device This field is set by the application to indicate that this channel is communicating to a low-speed device. |
| 16 | RO | 0x0 | reserved |
| 15 | RW | 0x0 | EPDir Endpoint Direction Indicates whether the transaction is IN or OUT. 1'b0: OUT 1'b1: IN |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31 | RW | 0x0 | SplitEna 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 15:14 | RW | 0x0 | <p>XactPos Transaction Position This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction.</p> <p>2'b11: All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes).</p> <p>2'b10: Begin. This is the first data payload of this transaction (which is larger than 188 bytes).</p> <p>2'b00: Mid. This is the middle payload of this transaction (which is larger than 188bytes).</p> <p>2'b01: End. This is the last payload of this transaction (which is larger than 188 bytes).</p> |
| 13:7 | RW | 0x00 | <p>HubAddr Hub Address This field holds the device address of the transaction translator's hub.</p> |
| 6:1 | RO | 0x0 | reserved |
| 0 | RW | 0x0 | <p>PrtAddr Port Address This field is the port number of the recipient transaction translator.</p> |

USBOTG_HCINTn

Address: Operational Base + offset (0x0508)

Host Channel-n Interrupt Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:14 | RO | 0x0 | reserved |
| 13 | W1C | 0x0 | <p>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.</p> |
| 12 | W1C | 0x0 | <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 | FrmOvrun Frame Overrun In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core |
| 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 9 | RW | 0x0 | FrmOvrunMsk Frame Overrun Mask This bit is not applicable in Scatter/Gather DMA mode. |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31 | RW | 0x0 | DoPng Do Ping 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. |

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:0 | RW | 0x00000000 | <p>DMAAddr DMA Address</p> <p>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.</p> |

USBOTG_HCDMABn

Address: Operational Base + offset (0x051c)

Host Channel-n DMA Buffer Address Register

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. |
| 25:24 | RW | 0x0 | 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. 2'b00: 25% of (micro)frame. 2'b01: 50% of (micro)frame. 2'b10: 75% of (micro)frame. 2'b11: Reserved. |

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 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: GAHBCFG.DMAEn=0, DCFG.DescDMA=0 => Slave mode GAHBCFG.DMAEn=0, DCFG.DescDMA=1 => Invalid GAHBCFG.DMAEn=1, DCFG.DescDMA=0 => Buffered DMA mode GAHBCFG.DMAEn=1, DCFG.DescDMA=1 => Scatter/Gather DMA mode</p> |
| 22:18 | RW | 0x08 | <p>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.</p> |
| 17:13 | RO | 0x0 | reserved |
| 12:11 | RW | 0x0 | <p>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%</p> |
| 10:4 | RW | 0x00 | <p>DevAddr Device Address The application must program this field after every SetAddress control command.</p> |
| 3 | RW | 0x0 | <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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) 2'b01: Full speed (USB 2.0 PHY clock is 30 MHz or 60 MHz) 2'b10: Reserved 2'b11: Full speed (USB 1.1 transceiver clock is 48 MHz)</p> |

USBOTG_DCTL

Address: Operational Base + offset (0x0804)

Device Control Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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> |

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 15 | RW | 0x0 | <p>IgnrFrmNum Ignore frame number for isochronous endpoints in case of Scatter/Gather DMA mode. 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>0: The core transmits the packets only in the frame number in which they are intended to be transmitted.</p> <p>1: 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. 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 microframes.</p> <p>0: Periodic transfer interrupt feature is disabled; the application must program transfers for periodic endpoints every (micro)frame.</p> <p>1: 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> |
| 14:13 | RW | 0x1 | <p>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.</p> <p>2'b00: Invalid. 2'b01: 1 packet. 2'b10: 2 packets. 2'b11: 3 packets.</p> <p>When Scatter/Gather DMA mode is disabled, this field is reserved. and reads 2'b00.</p> |
| 12 | RO | 0x0 | reserved |
| 11 | RW | 0x0 | <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 the 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. |
| 7 | WO | 0x0 | SGNPInNak 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. |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 1 | RW | 0x0 | <p>SftDiscon Soft Disconnect</p> <p>The application uses this bit to signal the DWC_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</p> <p>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 1ms 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

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 31:22 | RO | 0x0 | reserved |
| 21:8 | RW | 0x0000 | <p>SOFFN</p> <p>Frame or Microframe Number of the Received SOF</p> <p>When the core is operating at high speed, this field contains a microframe number. When the core is operating at full or low speed, this field contains a frame number.</p> |
| 7:4 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 3 | RW | 0x0 | <p>ErrticErr Erratic Error 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 DWC_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 Indicates the speed at which the DWC_otg core has come up after speed detection through a chirp sequence. 2'b00: High speed (PHY clock is running at 30 or 60 MHz) 2'b01: Full speed (PHY clock is running at 30 or 60 MHz) 2'b10: Low speed (PHY clock is running at 48 MHz, internal phy_clk at 6 MHz) 2'b11: Full speed (PHY clock is running at 48 MHz) Low speed is not supported for devices using a UTMI+ PHY.</p> |
| 0 | RW | 0x0 | <p>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).</p> |

USBOTG_DIEPMSK

Address: Operational Base + offset (0x0810)

Device IN Endpoint common interrupt mask register

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 6 | RW | 0x0 | INEPNakEffMsk IN Endpoint NAK Effective Mask |
| 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 |
| 2 | RW | 0x0 | AHBErrMsk AHB Error Mask |
| 1 | RW | 0x0 | EPDisbldMsk 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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 1 | RW | 0x0 | EPDisbldMsk Endpoint Disabled Interrupt Mask |
| 0 | RW | 0x0 | XferComplMsk Transfer Completed Interrupt Mask |

USBOTG_DAINT

Address: Operational Base + offset (0x0818)

Device All Endpoints interrupt register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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_DAINTMSK

Address: Operational Base + offset (0x081c)

Device All Endpoint interrupt mask register

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:28 | RO | 0x0 | reserved |
| 27 | RW | 0x1 | <p>ArbPrkEn Arbiter Parking Enable This bit controls internal DMA arbiter parking for IN endpoints. When thresholding 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 underrun conditions. By default the parking is enabled.</p> |
| 26 | RO | 0x0 | reserved |
| 25:17 | RW | 0x008 | <p>RxThrLen Receive Threshold Length This field specifies Receive thresholding 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. 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 DWORD 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. 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> |

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 10:2 | RW | 0x008 | <p>TxThrLen Transmit Threshold Length This field specifies Transmit thresholding 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 transmit 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.</p> <p>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 thresholding for isochronous IN endpoints.</p> |
| 0 | RW | 0x0 | <p>NonISOThrEn Non-ISO IN Endpoints Threshold Enable When this bit is set, the core enables thresholding for Non Isochronous IN endpoints.</p> |

USBOTG_DIEPEMPMSK

Address: Operational Base + offset (0x0834)

Device IN endpoint FIFO empty interrupt mask register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 31:16 | RO | 0x0 | reserved |
| 15:0 | RW | 0x0000 | <p>InEpTxfEmpMsk IN EP Tx FIFO Empty Interrupt Mask Bits These bits acts 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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 | EPDisbldMsk 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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 0 | RW | 0x0 | XferComplMsk Transfer Completed Interrupt Mask |

USBOTG_DIEPCTL0

Address: Operational Base + offset (0x0900)

Device control IN endpoint 0 control register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31 | R/W SC | 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 | R/W SC | 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. |
| 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. |

| Bit | Attr | Reset Value | Description |
|-------|-----------|-------------|---|
| 21 | R/W SC | 0x0 | <p>Stall STALL Handshake</p> <p>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.</p> |
| 20 | RO | 0x0 | reserved |
| 19:18 | RO | 0x0 | <p>EPType Endpoint Type</p> <p>Hardcoded to 00 for control</p> |
| 17 | RO | 0x0 | <p>NAKsts NAK Status</p> <p>Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status</p> <p>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.</p> |
| 16 | RO | 0x0 | reserved |
| 15 | RO | 0x1 | <p>USBActEP USB Active Endpoint</p> <p>This bit is always set to 1, indicating that control endpoint 0 is always active in all configurations and interfaces.</p> |
| 14:11 | RW | 0x0 | <p>NextEp Next Endpoint</p> <p>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.</p> <p>Note: This field is valid only for Shared FIFO operations.</p> |
| 10:2 | RO | 0x0 | reserved |
| 1:0 | RW | 0x0 | <p>MPS Maximum Packet Size</p> <p>Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint.</p> <p>2'b00: 64 bytes 2'b01: 32 bytes 2'b10: 16 bytes 2'b11: 8 bytes</p> |

USBOTG_DIEPINTn

Address: Operational Base + offset (0x0908)

Device Endpoint-n Interrupt Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 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 underrun condition for this endpoint. Dependency: This interrupt is valid only when both of the following conditions are true: Parameter OTG_ENDED_TX_FIFO==1; Thresholding 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. |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 7 | W1C | 0x0 | <p>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).</p> |
| 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. 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> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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. |
| 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_DIEPTSIZE_n

Address: Operational Base + offset (0x0910)

Device endpoint n transfer size register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31 | RO | 0x0 | reserved |
| 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> |
| 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 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.</p> |

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 18:0 | RW | 0x000000 | 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 coreConsultant 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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 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_n

Address: Operational Base + offset (0x0918)

Device IN endpoint transmit FIFO status register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|-------------|
| 31:16 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 15:0 | RW | 0x0000 | <p>INEPTxFSpcAvail IN Endpoint TxFIFO Space Avail Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words.</p> <p>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</p> |

USBOTG_DIEPDMAFn

Address: Operational Base + offset (0x091c)

Device endpoint-n DMA buffer address register

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:0 | RO | 0x00000000 | <p>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.</p> |

USBOTG_DIEPCTLn

Address: Operational Base + offset (0x0920)

Device endpoint-n control register

| Bit | Attr | Reset Value | Description |
|-----|--------|-------------|---|
| 31 | R/W SC | 0x0 | <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 30 | R/W SC | 0x0 | <p>EPDis Endpoint Disable</p> <p>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 | WO | 0x0 | <p>SetD1PID Set DATA1 PID</p> <p>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</p> <p>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 receive descriptor structure.</p> |
| 27 | WO | 0x0 | <p>SNAK Set NAK</p> <p>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.</p> |
| 26 | WO | 0x0 | <p>CNAK Clear NAK</p> <p>Applies to IN and OUT endpoints. A write to this bit clears the NAK bit for the endpoint.</p> |

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 25:22 | RW | 0x0 | <p>TxFNum Tx FIFO Number</p> <p>Shared FIFO Operation: non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic Tx FIFO number. 4'h0: Non-Periodic Tx FIFO; Others: Specified Periodic Tx FIFO 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.</p> <p>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.</p> |
| 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.</p> <p>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> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 17 | RO | 0x0 | <p>NAKSts NAK Status</p> <p>Applies to IN and OUT endpoints. Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>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 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> |
| 16 | RO | 0x0 | <p>DPID Endpoint Data PID</p> <p>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 SetD0PID 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> |

| Bit | Attr | Reset Value | Description |
|-------|--------|-------------|---|
| 15 | R/W SC | 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_DOEPCTL0

Address: Operational Base + offset (0x0b00)

Device control OUT endpoint 0 control register

| Bit | Attr | Reset Value | Description |
|-------|--------|-------------|---|
| 31 | R/W SC | 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. <i>Note: In DMA mode, this bit must be set for the core to transfer SETUP data packets into memory.</i> |
| 30 | WO | 0x0 | EPDis Endpoint Disable The application cannot disable control OUT endpoint 0. |
| 29:28 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|-------|--------|-------------|--|
| 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 | R/W SC | 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. |
| 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 |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 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 | BNAInrpt 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. |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 8 | W1C | 0x0 | <p>TxfifoUndrn FIFO Underrun</p> <p>Applies to IN endpoints only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. Dependency: This interrupt is valid only when both of the following conditions are true: Parameter OTG_EN_DED_TX_FIFO==1, Thresholding 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_EN_DED_TX_FIFO==1, Thresholding is enabled.</p> |
| 7 | W1C | 0x0 | <p>TxFEmp Transmit FIFO Empty</p> <p>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)).</p> |
| 6 | W1C | 0x0 | <p>INEPNakEff IN Endpoint NAK Effective</p> <p>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.</p> <p>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</p> <p>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)</p> <p>This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode.</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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. |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31 | RO | 0x0 | reserved |
| 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> |
| 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 during coreConsultant configuration (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.</p> <p>OUT Endpoints: This field is decremented every time a packet (maximum size or short packet) is written to the RxFIFO.</p> |

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 18:0 | RW | 0x000000 | 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 coreConsultant 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_DOEPDMAn

Address: Operational Base + offset (0x0b14)

Device Endpoint-n DMA Address Register

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31 | R/W SC | 0x0 | <p>EPEna Endpoint Enable</p> <p>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.</p> |
| 30 | R/W SC | 0x0 | <p>EPDis Endpoint Disable</p> <p>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</p> <p>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> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 receive descriptor structure.</p> |
| 27 | WO | 0x0 | <p>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.</p> |
| 26 | WO | 0x0 | <p>CNAK Clear NAK Applies to IN and OUT endpoints. A write to this bit clears the NAK bit for the endpoint.</p> |
| 25:22 | RW | 0x0 | <p>TxFNum Tx FIFO Number Shared FIFO Operation: non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic Tx FIFO number. 4'h0: Non-Periodic Tx FIFO; Others: Specified Periodic Tx FIFO 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.</p> |

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 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.</p> <p>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> |
| 17 | RO | 0x0 | <p>NAKSts NAK Status</p> <p>Applies to IN and OUT endpoints. Indicates the following:</p> <p>1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</p> <p>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 any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.</p> |

| Bit | Attr | Reset Value | Description |
|-------|--------|-------------|--|
| 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 SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>1'b0: DATA0 1'b1: DATA1 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> <p>1'b0: Even (micro)frame 1'b1: Odd (micro)frame 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 | R/W SC | 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> |
| 14:11 | RW | 0x0 | <p>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.</p> |
| 10:0 | RW | 0x000 | <p>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.</p> |

USBOTG_PCGCR

Address: Operational Base + offset (0x0b24)

Power and clock gating control register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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> <p>[31] if_dev_mode - 1: Device mode, core restored as device - 0: Host mode, core restored as host</p> <p>[30:29] p2hd_prt_spd (PRT speed) - 00: HS - 01: FS - 10: LS - 11: Reserved</p> <p>[28:27] p2hd_dev_enum_spd (Device enumerated speed) - 00: HS - 01: FS (30/60 MHz clk) - 10: LS - 11: FS (48 MHz clk)</p> <p>[26:20] mac_dev_addr (MAC device address) Device address</p> <p>[19] mac_termselect (Termination selection) - 0: HS_TERM (Program for High Speed) - 1: FS_TERM (Program for Full Speed)</p> <p>[18:17] mac_xcvrselect (Transceiver select) - 00: HS_XCVR (High Speed) - 01: FS_XCVR (Full Speed) - 10: LS_XCVR (Low Speed) - 11: LFS_XCVR (Reserved)</p> <p>[16] sh2pl_prt_ctl[0] - 1: prt_power enabled - 0: prt_power disabled</p> <p>[15:14] prt_clk_sel (Refer prt_clk_sel table)</p> |
| 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 |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 9 | RO | 0x0 | <p>RestoreMode Restore Mode (Applicable only when Hibernation is enabled (OTG_EN_PWROPT=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 |
| 3 | RW | 0x0 | <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 2 | RW | 0x0 | PwrClmp Power Clamp This bit is valid only in Partial Power-Down mode (OTG_EN_PWRDPT = 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. |

USBOTG_EPBUFO

Address: Operational Base + offset (0x1000)

Device endpoint 0 / host out channel 0 address

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RW | 0x00000000 | EPBUFO From 0x1000 to 0x2000, EPBUF for endport0 |

USBOTG_EPBUF1

Address: Operational Base + offset (0x2000)

Device endpoint 1 / host out channel 1 address

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:1 | RO | 0x0 | reserved |
| 0 | RW | 0x0 | EPBUF1 From 0x2000 to 0x3000, EPBUF for endport1 |

USBOTG_EPBUF2

Address: Operational Base + offset (0x3000)

Device endpoint 2 / host out channel 2 address

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:1 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 0 | RW | 0x0 | EPBUF2 From0x3000 to 0x4000, EPBUF for endport2 |

USBOTG_EPBUF3

Address: Operational Base + offset (0x4000)

Device endpoint 3 / host out channel 3 address

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:1 | RO | 0x0 | reserved |
| 0 | RW | 0x0 | EPBUF3 From0x4000 to 0x5000, EPBUF for endport3 |

USBOTG_EPBUF4

Address: Operational Base + offset (0x5000)

Device endpoint 4 / host out channel 4 address

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:1 | RO | 0x0 | reserved |
| 0 | RW | 0x0 | EPBUF4 From0x5000 to 0x6000, EPBUF for endport4 |

USBOTG_EPBUF5

Address: Operational Base + offset (0x6000)

Device endpoint 5 / host out channel 5 address

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:1 | RO | 0x0 | reserved |
| 0 | RW | 0x0 | EPBUF5 From0x6000 to 0x7000, EPBUF for endport5 |

USBOTG_EPBUF6

Address: Operational Base + offset (0x7000)

Device endpoint 6 / host out channel 6 address

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:1 | RO | 0x0 | reserved |
| 0 | RW | 0x0 | EPBUF6 From0x7000 to 0x8000, EPBUF for endport6 |

USBOTG_EPBUF7

Address: Operational Base + offset (0x8000)

Device endpoint 7 / host out channel 7 address

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:1 | RO | 0x0 | reserved |
| 0 | RW | 0x0 | EPBUF7 From0x8000 to 0x9000, EPBUF for endport7 |

2.7 Interface description

Table 5-21 USB OTG 2.0 Interface Description

| Module Pin | Direction | Pad Name | pinmux |
|------------|-----------|----------------|--------|
| USB0PN | A | IO_USB0_PN | - |
| USBRBIAS | A | IO_USB_RBIAS_0 | - |
| USB0PP | A | IO_USB0_PP | - |
| VBUS | A | IO_USB_VBUS_0 | - |
| USB0ID | A | IO_USB0_ID | - |

Note: **A**—Analog pad ; **AP**—Analog power; **AG**—Analog ground ;**DP**—Digital power ;**DG**—Digital ground;

2.8 Application Note

2.8.1 Suspend Mode

When PHY is in suspend state

- COMMONONNN = 1'b1, 480M clock invalid
- COMMONONNN = 1'b0, 480M clock output available.

Please refer to "Chapter GRF" for configuration details

2.8.2 Reset a port

Please refer to "Chapter CRU" for more details.

2.8.3 Relative GRF Registers

USBPHY contains 284 bit registers to configure USB PHY. These bits are used to adjust DP/DM SI. Please refer to "Chapter GRF" for more details.

Chapter 3 USB Host

3.1 Overview

There are three USB HOST controller in RK3228A/RK3228B. USB HOST0/1/2 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).

3.1.1 Features

- Compliant with the USB2.0 Specification
- Operates in Non_OTG Host mode
- Operates in High-Speed mode , Full-Speed, Low-speed mode
- Support 512x64 data buffer and 68x32 descriptor buffer .
- Support EHCI for High-Speed transfer, OHCI for Full-Speed, Low-speed transfer

3.2 Block Diagram

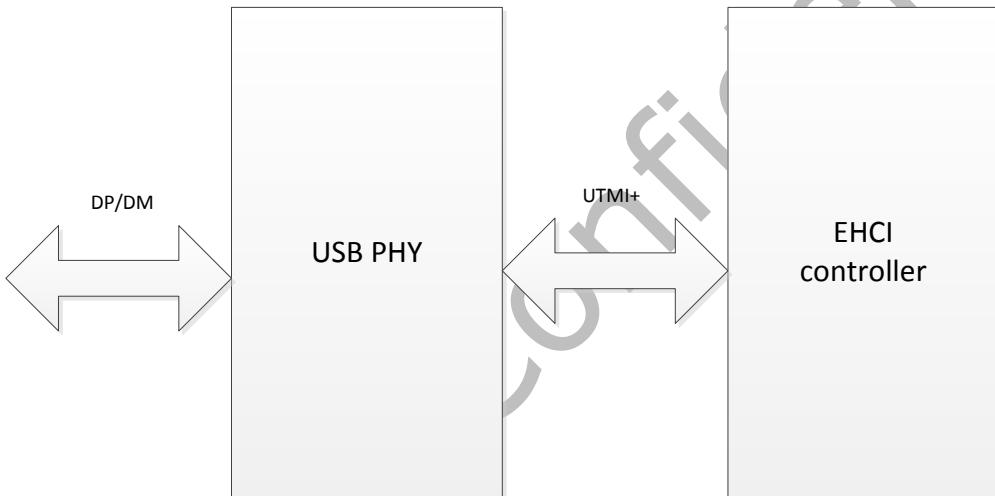


Fig. 5-21 USB HOST 2.0 Architecture

3.3 USB Host2.0 Controller

EHCI Controller is compliant with "Enhanced Host Controller Interface Specification for Universal Serial Bus". Please refer to "EHCI 1.0.pdf" for more information.

OHCI Controller is compliant with "Open Host Controller Interface specification". Please refer to "OHCI 1.0a.pdf" for more information.

3.4 USB Host2.0 PHY

Much the same as USB OTG PHY with no Device Mode supported. See Chapter OTG for more information.

USB Host2.0 PHY doesn't support UART-DEBUG function.

3.5 Register Description

The EHCI Controller Registers is the same as USB HSIC controller . Please refer Chapter HSIC for more information.

OHCI Controller is compliant with "OpenHCl specification". Please refer to "OpenHCl specification" for more information.

3.5.1 Internal Address Mapping

Slave address can be divided into different length for different usage, which is shown as follows.

| Name | Offset | Size | Reset Value | Description |
|----------------|--------|------|-------------|------------------------|
| HOST_INSNREG00 | 0x0090 | W | 0x00000020 | User-Specific Register |
| HOST_INSNREG01 | 0x0094 | W | 0x00001116 | User-Specific Register |
| HOST_INSNREG02 | 0x0098 | W | 0x00000000 | User-Specific Register |
| HOST_INSNREG03 | 0x009c | W | 0x00000000 | User-Specific Register |
| HOST_INSNREG04 | 0x00a0 | W | 0x00000000 | User-Specific Register |
| HOST_INSNREG05 | 0x00a4 | W | 0x00001000 | User-Specific Register |
| HOST_INSNREG06 | 0x00a8 | W | 0x00000000 | User-Specific Register |
| HOST_INSNREG07 | 0x00ac | W | 0x00000000 | User-Specific Register |

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

3.5.2 Registers Summary

| Name | Offset | Size | Reset Value | Description |
|----------------|--------|------|-------------|------------------------|
| HOST_INSNREG00 | 0x0090 | W | 0x00000020 | User-Specific Register |
| HOST_INSNREG01 | 0x0094 | W | 0x00001116 | User-Specific Register |
| HOST_INSNREG02 | 0x0098 | W | 0x00000000 | User-Specific Register |
| HOST_INSNREG03 | 0x009c | W | 0x00000000 | User-Specific Register |
| HOST_INSNREG04 | 0x00a0 | W | 0x00000000 | User-Specific Register |
| HOST_INSNREG05 | 0x00a4 | W | 0x00001000 | User-Specific Register |
| HOST_INSNREG06 | 0x00a8 | W | 0x00000000 | User-Specific Register |
| HOST_INSNREG07 | 0x00ac | W | 0x00000000 | User-Specific Register |

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

3.5.3 Detail Register Description

HOST_INSNREG00

Address: Operational Base + offset (0x0090)

User-Specific Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 31:14 | RO | 0x0 | reserved |
| 13:1 | RW | 0x0010 | <p>sof_scaledown Programmable Microframe Base Value Allows you to change the microframe length value (default is microframe SOF = 125 µs) to reduce the simulation time. [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).</p> |
| 0 | RW | 0x0 | <p>insnreg00_enable Writing 1 enables this register. Note: Do not enable this register for the gate-level netlist.</p> |

HOST_INSNREG01

Address: Operational Base + offset (0x0094)

User-Specific Register

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:0 | RW | 0x00001116 | <p>in_out_threshold Programmable Packet Buffer OUT/IN Thresholds (in CONFIG1 mode only, not applicable in Config2 mode).</p> <p>[31:16] OUT Threshold [15:0] IN 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.</p> <p>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.</p> <p>The minimum OUT and IN threshold amount that can be programmed through INSN registers is 16 bytes.</p> <p>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> <ol style="list-style-type: none"> 1. Packet buffer depth is equal to 512 bytes and isochronous/interrupt transactions are not initiated by the host controller. 2. Packet buffer depth is equal to 1024 bytes. <p>The default value of thresholds depend on one of the following packet buffer configurations</p> <ul style="list-style-type: none"> 1. 1024 bytes depth, 256 bytes IN and OUT thresholds 2. 512 bytes depth, 128 bytes IN and OUT thresholds 3. 256 bytes depth, 64 bytes IN and OUT thresholds 4. 128 bytes depth, 64 bytes IN and OUT thresholds <p>For INCRX configurations, the Break Memory Transfer bit is always enabled.</p> |

HOST_INSNREG02

Address: Operational Base + offset (0x0098)

User-Specific Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|-------------|
| 31:12 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 11:0 | RW | 0x010 | insnreg02 Programmable Packet Buffer Depth (in CONFIG1 mode only, not applicable in CONFIG2 mode) 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. |

HOST_INSNREG03

Address: Operational Base + offset (0x009c)

User-Specific Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:14 | RO | 0x0 | reserved |
| 13 | RW | 0x0 | ignore_ls_in_testse0nak 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 behaviour during this. |
| 12:10 | RW | 0x0 | turnaround_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. |
| 9 | RW | 0x0 | periodic_frame_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 microframe of a frame. If not set, then the periodic frame list will be fetched only in microframe 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. |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 8:1 | RW | 0x00 | <p>time_available_offset 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 | 0x1 | <p>break_memory_transfer [0] 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> |

HOST_INSNREG04

Address: Operational Base + offset (0x00a0)

User-Specific Register

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:6 | RO | 0x0 | reserved |
| 5 | RW | 0x0 | <p>suspend_auto_en 1'b0: 0 by default, the automatic feature is enabled. The Suspend signal is deasserted (logic level 1'b1) when run/stop is reset by software, but the hhalted 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 in release 2.80a and later. 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 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.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 4 | RW | 0x0 | nak_reload_fix_enable 1'b0: NAK reload fix enabled. 1'b1: NAK reload fix disabled. (Incorrect NAK reload transition at the end of a microframe for backward compatibility with Release 2.40c.) |
| 3 | RO | 0x0 | reserved |
| 2 | RW | 0x0 | scales_down_enum_time When 1'b1, Scales down port enumeration time. Reset value is 1'b0. |
| 1 | RW | 0x0 | hcsparms_wr_enable1 When 1'b1 ,The HCCPARAMS register's bits 17, 15:4, and 2:0 become writable. Upon system reset, these bits are 0. |
| 0 | RW | 0x0 | hcsparms_wr_enable0 When 1'b1, The HCSPARAMS register becomes writable. Upon system reset, this bit is 0. |

HOST_INSNREG05

Address: Operational Base + offset (0x00a4)

User-Specific Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:18 | RO | 0x0 | reserved |
| 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 Valid values range from 1 to 15 depending on coreConsultant configuration. 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 | 0x1 | VControlLoadM 1'b0: Load 1'b1:NOP |
| 11:8 | RW | 0x0 | vcontrol VControl control bits. |
| 7:0 | RO | 0x00 | vstatus VSTATUS value |

HOST_INSNREG06

Address: Operational Base + offset (0x00a8)

User-Specific Register

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

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 31 | RW | 0x0 | ahb_error_cap Indicator that an AHB error was encountered and values were captured. To clear this field the application must write a 0 to it. |
| 30:12 | RO | 0x0 | reserved |
| 11:9 | RO | 0x0 | hburst HBURST Value of the control phase at which the AHB error occurred. |
| 8:4 | RO | 0x00 | beats_num Number of beats expected in the burst at which the AHB error occurred. Valid values are 0 to 16. - 5'b10001 磡b11111: Reserved - 5'b00000 磡b10000: Valid |
| 3:0 | RO | 0x0 | succ_beats_num Number of successfully-completed beats in the current burst before the AHB error occurred. |

HOST_INSNREG07

Address: Operational Base + offset (0x00ac)

User-Specific Register

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:0 | RO | 0x00000000 | error_addr AHB Master Error Address AHB address of the control phase at which the AHB error occurred |

3.6 Interface description

Table 5-22USB HOST 2.0 Interface Description

| Module Pin | Direction | Pad Name | Description |
|------------|-----------|------------|---------------------|
| USB1PN | A | IO_USB1_PN | Host 0 USB data bus |
| USB1PP | A | IO_USB1_PP | Host 0 USB data bus |
| USB0PN | A | IO_USB2_PN | Host 1 USB data bus |
| USB0PP | A | IO_USB2_PP | Host 1 USB data bus |
| USB1PN | A | IO_USB3_PN | Host 2 USB data bus |
| USB1PP | A | IO_USB3_PP | Host 2 USB data bus |

Note: **A**—Analog pad ; **AP**—Analog power; **AG**—Analog ground ; **DP**—Digital power ; **DG**— Digital ground;

3.7 Application Note

See Chapter OTG for more information.

3.7.1 Reset a port

Please refer to "Chapter CRU" for more details.

3.7.2 Relative GRF Registers

Please refer to "Chapter GRF" for more details.

Chapter 4 HDMI TX

4.1 Overview

HDMI TX is fully compliant with HDMI 1.4a and 2.0 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 (configuration dependent)
 - ◆ All CEA-861-E video formats up to 1080p at 120 Hz
 - ◆ HDMI 1.4b 4K x 2K video formats
 - ◆ HDMI 1.4b 3D video modes with up to 340 MHz (TMDS clock)
 - HDMI 2.0 video formats, All CEA-861-F video formats
- Colorimetry, 24-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 YCbCr420 10bit
- Integrated CEC hardware engine

4.2 Block Diagram

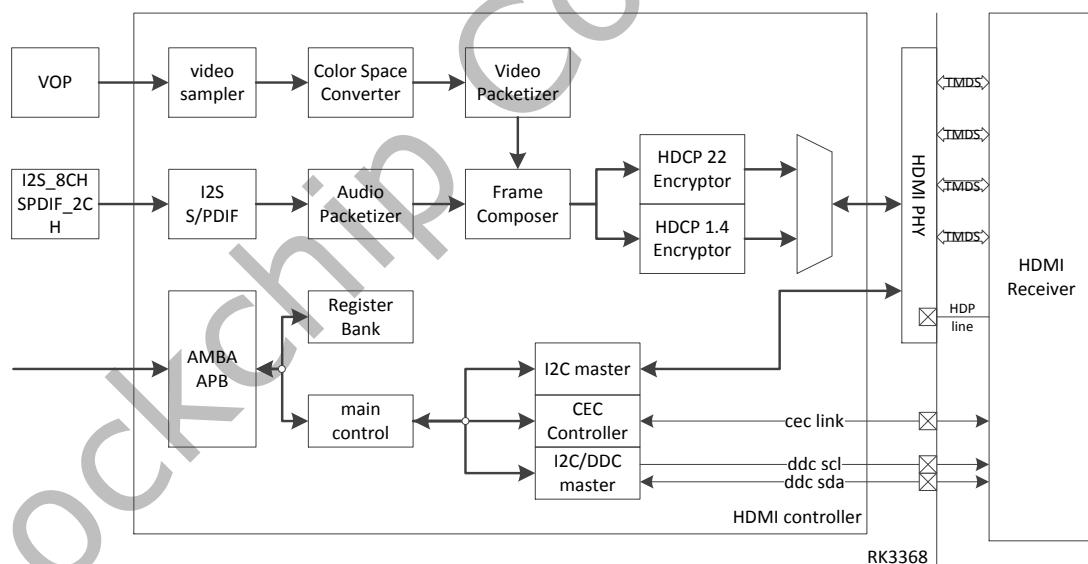


Fig. 5-22 HDMI TX Block Diagram

4.3 Function Description

4.3.1 IOMUX

Customer should notice that CEC channel use IO_HDMICECSDA_GPIO0c4 and DDC channel use IO_I2C3scl_HDMIddcscl_GPIO0a6 and IO_I2C3sda_HDMIddcsda_GPIO0a7 in 3v mode, or IO_I2C3scl_HDMIddcscl_GPIO0a6_5v and IO_I2C3sda_HDMIddcsda_GPIO0a7_5v in 5v mode.

The hpd use IO_HDMIhpd_GPIO0b7 in 3v mode or IO_HDMIhpd_GPIO0b7_5v in 5v mode.

4.3.2 Pixel Clock

When using HDMI as display solution, customers can set pixel clock source to CRU or HDMI PHY PCLK for VOP and HDMI controller. Here HDMI PHY PCLK is recommended.

4.3.3 Video Pixel Sampler

In RK3228A/RK3228B, HDMI data comes from VOP. The Video pixel sampler block is responsible for the video data synchronization, according to the video data input mapping defined by the Color Depth (Deep Color) and format configuration. Register must be set correctly or cohesively with VOP output.

4.3.4 Video Data Processing

The video processing contains 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. 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 24bits. 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.

Color Space Conversion

HDMI Transmitter Color space conversion (CSC) is responsible for carrying out the following video color space conversion functions:

- RGB to/from YCbCr
- 4:2:2 to/from 4:4:4 up (pixel repetition or linear interpolation)/down-converter
- Limited to/from full quantization range conversion
- The CSC supports all the timings reported in the CEA-861-D specification and the following pixel modes:
- RGB 444 and YCbCr 444: 24, 30, 36, and 48 bits
- YCbCr 422: 16, 20, and 24 bits

The color space conversion matrix is ruled by the following equations listed in below figure.

$$\begin{aligned} \text{out}_1 &= (X_1 \times \text{in}_1 / 4096 + X_2 \times \text{in}_2 / 4096 + X_3 \times \text{in}_3 / 4096 + X_4) \times 2^{\text{scale}} \\ \text{out}_2 &= (Y_1 \times \text{in}_1 / 4096 + Y_2 \times \text{in}_2 / 4096 + Y_3 \times \text{in}_3 / 4096 + Y_4) \times 2^{\text{scale}} \\ \text{out}_3 &= (Z_1 \times \text{in}_1 / 4096 + Z_2 \times \text{in}_2 / 4096 + Z_3 \times \text{in}_3 / 4096 + Z_4) \times 2^{\text{scale}} \end{aligned}$$

Fig. 5-23 HDMI Color Space Conversion Matrix Equations

Note: Color Space Conversion to and from YCrCb 4:2:0 is not supported.

4.3.5 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:

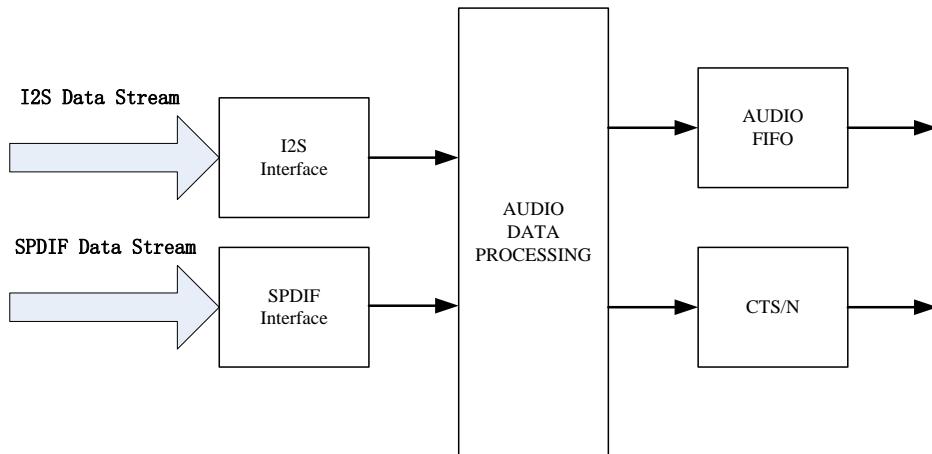


Fig. 5-24 HDMI Audio Data Processing Diagram

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 5-23 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 5-24 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 5-25 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/ | Yes | Yes | Yes | Yes | Yes | No | No |

| | | | | | | | |
|-----------|-----|-----|-----|-----|-----|-----|-----|
| 1440x576i | | | | | | | |
| 720p | Yes |
| 1080i | Yes |
| 1080p | Yes |

3.Audio Sample Clock Capture and Regeneration

Audio data is 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.

$$128 \cdot f_s = f_{\text{TMDS_clock}} * N / CTS.$$

The scheme of the Audio Sample Clock Capture and Regeneration as shown below:

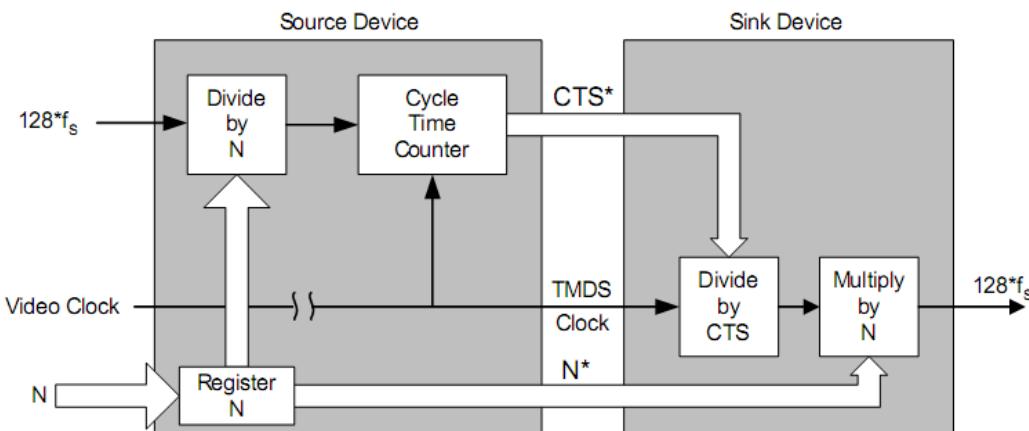


Fig. 5-25 HDMI Audio Clock Regeneration Model

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 form 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 5-26 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 | N |
| 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 |

4.3.6 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. DDC bus access frequency can be controlled. I2C input can be masked to zero by clear relative bits in grf_soc_con2,grf_soc_con2[13] mask i2c_scl while grf_soc_con2[14] mask i2c_sda.

4.3.7 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.

4.3.8 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.

RK3228A/RK3228B supports up to HDCP 2.2. HDCP 1.4 is done by HDMI Transmitter itself. HDCP22 is described in next chapter.

4.3.9 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.

4.3.10 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.

4.3.11 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.

4.4 HDMI PHY

The HDMI Tx PHY is the physical layer of an HDMI digital transmitter (source), capable of encoding and transmitting high-speed data streams carrying RGB video, audio, and control information.

The HDMI TX PHY is compliant with HDMI2.0 specification and optimized up to 3.72Gbps per TMDS link High-Definition applications which supporting 3D display and up to 4Kx2K@60/50Hz Ultra High Definition(UHD) resolution at 10-bit YUV420 video input mode. The HDMI TX PHY contains programmable circuit with pre-emphasis, slew rate, output voltage swing and resistance values to drive transmit data with optimal signal quality and distance over the standard HDMI cable to a stand HDMI display device. The TX PHY contains all HDMI components including all I/O library, high performance wide range PLL, the data symbol conversion/synchronization unit.

Below is the TX PHY features:

- HDMI 2.0/1.4a/b/1.3/1.2/1.1,DVI 1.0 standard compliant transmitter PHY
- Supports data rate from 25M, up to 3.72Gbps over a single TMDS channel
- Supports TMDS TX with programmable output swing, resistance values and pre-emphasis
- Low power consumption with less than 100mW for PHY in UHD 4K displaying mode
- Library delivered to support all major EDA tools and detailed guide to integrate into pad ring/BGA bumps
- 1.8V high speed I/O and 1.0V core power supply
- APB slave interface for internal registers access
- GF28nm SLP process ready with die size less than 0.6mm² before shrink for PHY(including IO PADs)

4.4.1 ARCHITECTURE

The TX PHY integrates low jitter PLL, bandgap reference and 4 TMDS channel to deliver multimedia data. Specialized transmitter driver circuit with tunable pre-emphasis, slew rate, and output voltage swing is able to transmit data with optimal signal quality and distance over

the standard HDMI cable. Basically, the output driver switches the current supplied by the current source and drives the differential pair line.

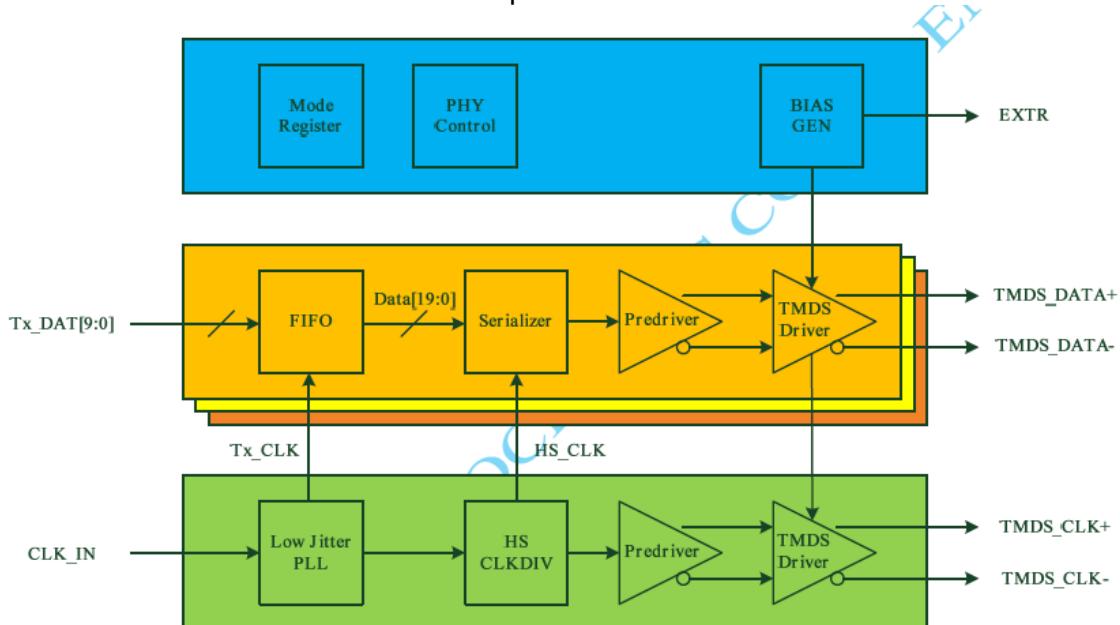


Fig. 5-26 Sample INNO HDMI2.0 TX PHY Block Diagram

4.4.2 Power Control Timing

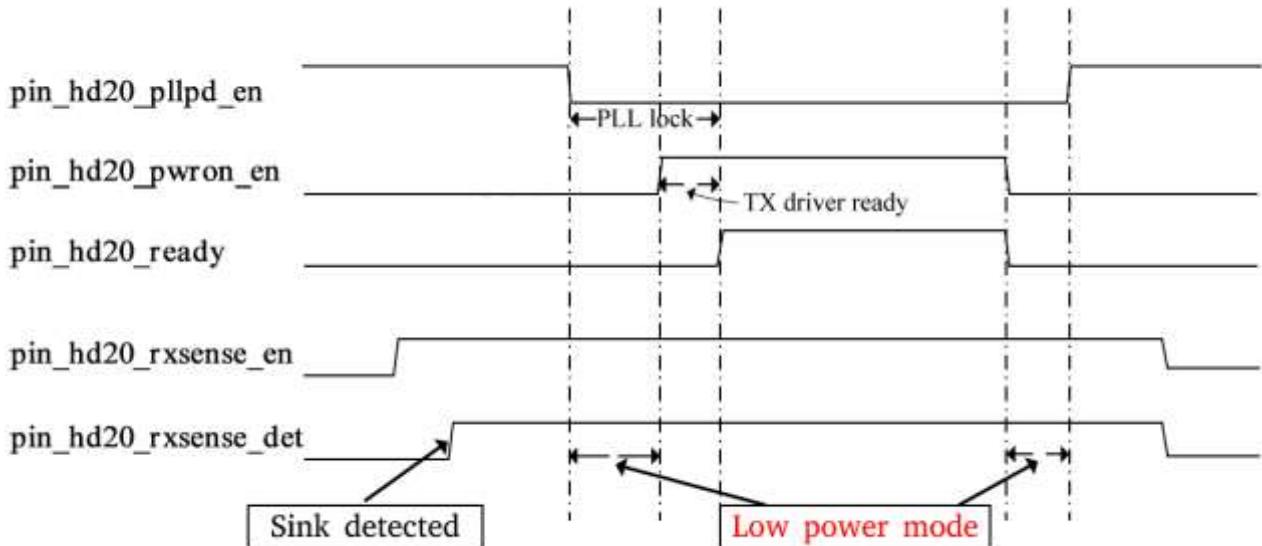


Fig. 5-27 Power relative signals timing

Pin_hd20_pwrone_en is the HDMI PHY tx driver's power enable and analog reset signal, which is configured from grf. After pll is power up and ready, this signal must be set from 0 to 1 to reset HDMI PHY tx analog driver. pin_hd20_pllpd_en is PLL power down signal, the PLL is powered down when this pin set high. It is recommended to power down the driver and pll before app bus configuration, and do not pull the pin_hd20_pwrone_en high until pin_hd20_ready rise.

4.4.3 PLL DESCRIPTION

There are two PLLs in TX PHY: PRE-PLL and POST-PLL.

The PRE-PLL is used to generate tmds clk, tmds character clk, pclk and prepclk. The POST-PLL is only used to generate differential serial clock for drivers which differential frequency is always 5 times of the pin_hd20_tmdsclk.

You should make sure the VCO frequency at range 750Mhz~3Ghz for both PRE-PLL and POST-PLL.

4.4.4 PHY Registers Summary

| Name | Offset | Size | Reset Value | Description |
|---|--------|------|-------------|---|
| HDMI_PHY_general_reg | 0x0000 | W | 0x000000c0 | |
| HDMI_PHY_termination_res_cal | 0x000c | W | 0x00000083 | By pass auto termination resistor calibration |
| HDMI_PHY_termination_res_speed | 0x0010 | W | 0x00000000 | |
| HDMI_PHY_prepll_conf | 0x0388 | W | 0x00000000 | prepll configuration |
| HDMI_PHY_prepll_fb_div | 0x038c | W | 0x00000000 | prepll feedback divider bit[7:0] |
| HDMI_PHY_prepll_fb_div_ab | 0x0390 | W | 0x00000000 | |
| HDMI_PHY_prepll_fb_div_cd | 0x0394 | W | 0x00000000 | |
| HDMI_PHY_tmds_divider | 0x0398 | W | 0x00000000 | |
| HDMI_PHY_postpll_div_en | 0x03a4 | W | 0x000000e4 | postpll divider enable |
| HDMI_PHY_postpll_fb_div | 0x03a8 | W | 0x00000000 | |
| HDMI_PHY_postpll_fb_div2 | 0x03ac | W | 0x00000000 | |
| HDMI_PHY_tmds_adj_en | 0x03b8 | W | 0x0000000f | |
| HDMI_PHY_tmds_adj_sel | 0x03bc | W | 0x0000002a | |
| HDMI_PHY_preamphasis_level_ctrl | 0x03c0 | W | 0x00000000 | |
| HDMI_PHY_tmds_output_swинг_con0 | 0x03c4 | W | 0x00000000 | |
| HDMI_PHY_tmds_output_swинг_conf1 | 0x03c8 | W | 0x00000000 | |
| HDMI_PHY_tmds_termination_res_cal | 0x03d0 | W | 0x0000000c | |
| HDMI_PHY_tmds_clock_termination_res_setting | 0x03ec | W | 0x00000000 | |
| HDMI_PHY_tmds_d2_termination_res_setting | 0x03f0 | W | 0x00000000 | |
| HDMI_PHY_tmds_d1_termination_res_setting | 0x03f4 | W | 0x00000000 | |
| HDMI_PHY_tmds_d0_termination_res_setting | 0x03f8 | W | 0x00000000 | |

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

4.4.5 Detail PHY Register Description

HDMI_PHY_general_reg

Address: Operational Base + offset (0x0000)

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:8 | RO | 0x0 | reserved |
| 7 | RW | 0x1 | analog_rst analog resetn 0: reset |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 6 | RW | 0x1 | dig_RST soft digital reset_n 0: reset |
| 5:4 | RO | 0x0 | reserved |
| 3 | RW | 0x0 | pclk_inv_en pclk invert enable 1: invert 0: not invert |
| 2 | RW | 0x0 | prepclk_inv_en prepclk invert enable 1: invert 0: not invert |
| 1 | RW | 0x0 | tmdsclk_inv_en tmdsclk invert enable 1: invert 0: not invert |
| 0 | RW | 0x0 | prepll_ref_sel pre-PLL reference clock select 0: select synchronous pclk 1: sel crystal oscillator clock |

HDMI_PHY_termination_res_cal

Address: Operational Base + offset (0x000c)

By pass auto termination resistor calibration

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:8 | RO | 0x0 | reserved |
| 7:1 | RW | 0x41 | Reserve |
| 0 | RW | 0x1 | terminitaion_res_cal By pass auto termination resistor calibration 1: bypass 0: not bypass |

HDMI_PHY_termination_res_speed

Address: Operational Base + offset (0x0010)

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:8 | RO | 0x0 | reserved |
| 7:0 | RW | 0xe8 | termination_res_speed auto termination resistor calibration speed control Freq(configure control) = Freq(sys_clk)/reg10[6:0],reg14[[7:0]. Typically 100khz. Assume sys_clk is 100Mhz, then this value should be set to 100Mhz/100Khz = 1000. Notes: this value only effective when not bypass auto termination resistor calibration. |

HDMI_PHY_preppll_conf

Address: Operational Base + offset (0x0388)

preppll configuration

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:8 | RO | 0x0 | reserved |
| 7 | RW | 0x0 | preppll_fb_div preppll feedback divider bit[8] |
| 6 | RO | 0x0 | reserved |
| 5 | RW | 0x0 | pclk_div_5_en enable of pclk directly divided by 5 from VCO of the preppll 1: enable 0: disable |
| 4:0 | RW | 0x02 | preppll_div preppll pre divider[4:0] |

HDMI_PHY_preppll_fb_div

Address: Operational Base + offset (0x038c)

preppll feedback divider bit[7:0]

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:8 | RO | 0x0 | reserved |
| 7:0 | RW | 0x63 | preppll_fb_div preppll feedback divider bit[7:0] Field0000 Description |

HDMI_PHY_preppll_fb_div_ab

Address: Operational Base + offset (0x0390)

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:7 | RO | 0x0 | reserved |
| 6:5 | RW | 0x0 | pclk_dividerb preppll pclk_dividerb 00: divide by 2 01: divide by 3 10: divide by 4 11: divide by 5 |
| 4:0 | RW | 0x00 | pclk_dividera preppll pclk-dividera Range from 1 to 31 |

HDMI_PHY_preppll_fb_div_cd

Address: Operational Base + offset (0x0394)

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:7 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 6:5 | RW | 0x2 | pclk_dividerc prepll pclk_dividerc 00: divide by 1 01: divide by 2 10: divide by 4 11: divide by 8 |
| 4:0 | RW | 0x02 | pclk_dividerd prepll pclk dividerd Range from 1 to 31 |

HDMI_PHY_tmds_divider

Address: Operational Base + offset (0x0398)

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:6 | RO | 0x0 | reserved |
| 5:4 | RW | 0x0 | tmds_dividerc prepll tmds dividerc 00: divide by 1 01: divide by 2 10: divide by 4 11: divide by 8 |
| 3:2 | RW | 0x1 | tmds_dividera prepll tmds dividera 00: divide by 1 01: divide by 2 10: divide by 3 11: divide by 5 |
| 1:0 | RW | 0x0 | tmds_dividerb prepll tmds dividerb 00: divide by 1 01: divide by 2 10: divide by 4 11: divide by 8 |

HDMI_PHY_postpll_div_enAddress: Operational Base + offset (0x03a4)
postpll divider eanble

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:8 | RO | 0x0 | reserved |
| 7 | RW | 0x1 | Reserv |
| 6:5 | RW | 0x3 | post_div_en post pll divider enable 2'b11: enable 2'b00: disable others: invalid |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 4:0 | RW | 0x04 | postpll_pre_divider Field0000 Abstract Field0000 Description |

HDMI_PHY_postpll_fb_div

Address: Operational Base + offset (0x03a8)

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:8 | RO | 0x0 | reserved |
| 7:0 | RW | 0x50 | postpll_fb_div post pll feedback divider bit[7:0] |

HDMI_PHY_postpll_fb_div2

Address: Operational Base + offset (0x03ac)

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:8 | RO | 0x0 | reserved |
| 7 | RW | 0x0 | postpll_fb_div_bit8 |
| 6:5 | RO | 0x0 | reserved |
| 4 | RW | 0x0 | postpll_div 2'bx0:divider by 2 2'b01:divider by 4 2'b11:divider by 8 Note: post divider only effective when post pll post divider enable is set. |
| 3:1 | RO | 0x0 | reserved |
| 0 | RW | 0x0 | postpll_lock post pll lock status 1: locked 0: unlock |

HDMI_PHY_tmds_adj_en

Address: Operational Base + offset (0x03b8)

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:8 | RO | 0x0 | reserved |
| 7 | RW | 0x0 | tmds_clk_tr_adj Transmition adjustment enable for TMDS clock channel 1: enable 0: disable |
| 6 | RW | 0x0 | tmds_d0_tr_adj Transmition adjustment enable for TMDS data channel 0 1: enable 0: disable |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 5 | RW | 0x0 | tmds_d1_tr_adj Transmition adjustment enable for TMDS data channel 1 1: enable 0: disable |
| 4 | RW | 0x0 | tmds_d2_tr_adj Transmition adjustment enable for TMDS data channel 2 1: enable 0: disable |
| 3 | RW | 0x1 | tmds_clk_cur_ctl switch current control for tmds clock channel driver 1: enable 0: disable |
| 2 | RW | 0x1 | tmds_d0_cur_ctl switch current control for tmds data channel 0 driver 1: enable 0: disable |
| 1 | RW | 0x1 | tmds_d1_cur_ctl switch current control for tmds data channel 1 driver 1: enable 0: disable |
| 0 | RW | 0x1 | tmds_d2_cur_ctl switch current control for tmds data channel 2 driver 1: enable 0: disable |

HDMI_PHY_tmds_adj_sel

Address: Operational Base + offset (0x03bc)

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:8 | RO | 0x0 | reserved |
| 7:6 | RW | 0x2 | tmds_clk_tr_adj_sel Transition adjustment for tmds clock channel 00: slowest rising and failing edge 01: slow rising and failing edge 10: fast rising and failing edge 11: fastest rising and falling edge |
| 5:4 | RW | 0x2 | tmds_d2_tr_adj_sel Transition adjustment for tmds data channel 2 00: slowest rising and failing edge 01: slow rising and failing edge 10: fast rising and failing edge 11: fastest rising and falling edge |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 3:2 | RW | 0x2 | <p>tmds_d1_tr_adj_sel Transition adjustment for tmds data channel 1 00: slowest rising and failing edge 01: slow rising and failing edge 10: fast rising and failing edge 11: fastest rising and falling edge</p> |
| 1:0 | RW | 0x2 | <p>tmds_d0_tr_adj_sel Transition adjustment for tmds data channel 0 00: slowest rising and failing edge 01: slow rising and failing edge 10: fast rising and failing edge 11: fastest rising and falling edge</p> |

HDMI_PHY_preemphassis_level_ctrl

Address: Operational Base + offset (0x03c0)

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:4 | RO | 0x0 | reserved |
| 3 | RW | 0x0 | <p>tmds_clk_preemphasis Pre-emphasis level control for TMDS clock channel 00: disable pre-emphasis 01: small pre-emphasis level 10: moderate pre-emphasis level 11: large pre-emphasis level</p> |
| 2 | RW | 0x0 | <p>tmds_d2_preemphasis Pre-emphasis level control for TMDS data channel 2 00: disable pre-emphasis 01: small pre-emphasis level 10: moderate pre-emphasis level 11: large pre-emphasis level</p> |
| 1 | RW | 0x0 | <p>tmds_d1_preemphasis Pre-emphasis level control for TMDS data channel 1 00: disable pre-emphasis 01: small pre-emphasis level 10: moderate pre-emphasis level 11: large pre-emphasis level</p> |
| 0 | RW | 0x0 | <p>tmds_d0_preemphasis Pre-emphasis level control for TMDS data channel 0 00: disable pre-emphasis 01: small pre-emphasis level 10: moderate pre-emphasis level 11: large pre-emphasis level</p> |

HDMI_PHY_tmds_output_swing_con0

Address: Operational Base + offset (0x03c4)

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:8 | RO | 0x0 | reserved |
| 7:4 | RW | 0x4 | tmds_clk_output_swing output swing control for tmds clock channel 0: minimal swing 15: maximal swing |
| 3:0 | RW | 0x4 | tmds_d2_output_swing output swing control for tmds data channel 2 0: minimal swing 15: maximal swing |

HDMI_PHY_tmds_output_swing_conf1

Address: Operational Base + offset (0x03c8)

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:8 | RO | 0x0 | reserved |
| 7:4 | RW | 0x4 | tmds_d1_output_swing output swing control for tmds data channel 1 0: minimal swing 15: maximal swing |
| 3:0 | RW | 0x4 | tmds_d0_output_swing output swing control for tmds data channel 2 0: minimal swing 15: maximal swing |

HDMI_PHY_tmds_termination_res_cal

Address: Operational Base + offset (0x03d0)

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:4 | RO | 0x0 | reserved |
| 3:2 | RW | 0x3 | tmds_termination_res_cal auto termination resistor calibration setting for all tmds channels 00: 50Ohm 01: 75Ohm 10: 100Ohm 11: open |
| 1:0 | RO | 0x0 | reserved |

HDMI_PHY_tmds_clock_termination_res_setting

Address: Operational Base + offset (0x03ec)

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:6 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 5:0 | RW | 0x00 | tmds_clk_termination_res_setting Manual termination resistor setting for driver of tmds clock channel Only effective when bypass auto termination resistor calibration |

HDMI_PHY_tmds_d2_termination_res_setting

Address: Operational Base + offset (0x03f0)

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:6 | RO | 0x0 | reserved |
| 5:0 | RW | 0x00 | tmds_d2_termination_res_setting Manual termination resistor setting for driver of tmds data channel2 Only effective when bypass auto termination resistor calibration |

HDMI_PHY_tmds_d1_termination_res_setting

Address: Operational Base + offset (0x03f4)

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:6 | RO | 0x0 | reserved |
| 5:0 | RW | 0x00 | tmds_d1_termination_res_setting Manual termination resistor setting for driver of tmds data channel1 Only effective when bypass auto termination resistor calibration |

HDMI_PHY_tmds_d0_termination_res_setting

Address: Operational Base + offset (0x03f8)

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:6 | RO | 0x0 | reserved |
| 5:0 | RW | 0x00 | tmds_d0_termination_res_setting Manual termination resistor setting for driver of tmds data channel0 Only effective when bypass auto termination resistor calibration |

4.5 Register Description

The base address of the HDMI TX is 0x200a0000, 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 offset address is 0x0100. If we want to configure it, its real address is 0x200a0000+0x0100*4.

4.5.1 Register Summary

| Name | Offset | Size | Reset Value | Description |
|--------------------------|--------|------|-------------|----------------------------------|
| Identification Registers | 0x0000 | B | | Identification related registers |

| Name | Offset | Size | Reset Value | Description |
|--|--------|------|-------------|--|
| 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 Synopsys's doc:
DWC_hdmi_tx_databook.pdf.

4.6 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.

4.6.1 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.

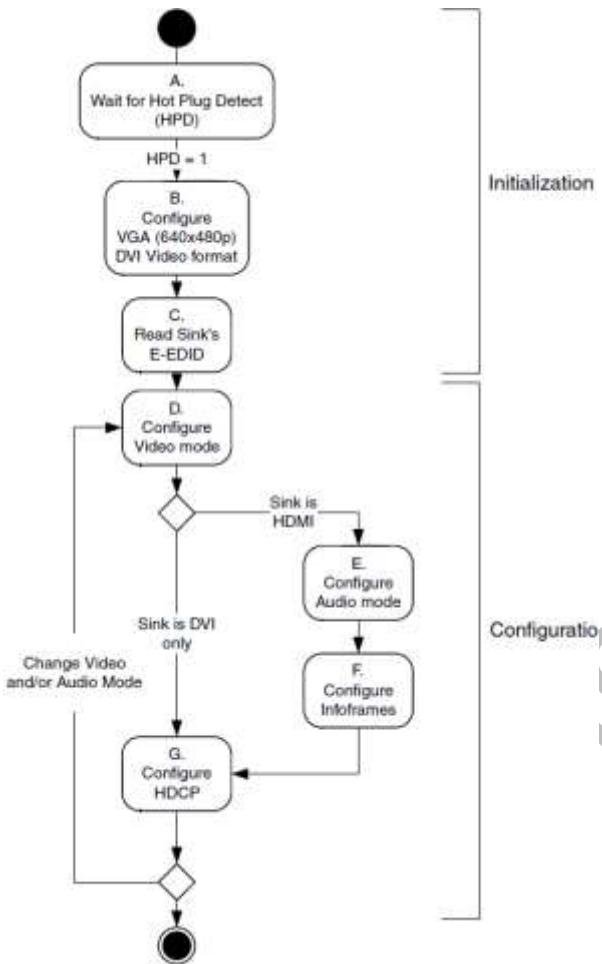


Fig. 5-28 HDMI TX Block Diagram

4.6.2 Step A: Hot Plug Detection

The Hot Plug Detect information notifies the HDMI TX controller when the cable is plugged into a Sink (Receiver) device and when the device is ready to receive video content.

The Hot-Plug Detection indication is a +5V signal on the HDMI cable. The HDMI TX PHY performs level-shifting to a low-level digital signal.

For this step, registers are used on the HDMI TX PHY.

To check the Hot Plug Detect status, the following steps have to be followed:

1. Power-on the HDMI TX PHY HPD Detector. Write 1'b1 in the phy_conf0.enhpdrxsense bit field register.
2. Check the HPD status bit.
 - 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).

Note: Some receivers may turn off the Hot-Plug Detect indication when the HDMI input is not selected, even when the cable is plugged in.

You can control this function by using the interrupt signal and proper registers from the HDMI transmitter with few operations.

4.6.3 Step B: Configure Video Mode

Once a receiver is detected, it is necessary to start sending video content in DVI mode.

Because some receivers need to receive a video signal (TMDS clock more precisely), it is important to clock all the digital logic and be able to reply on an E-EDID read access (Step C).

Following flow is a configuration example of VGA (640x480p).

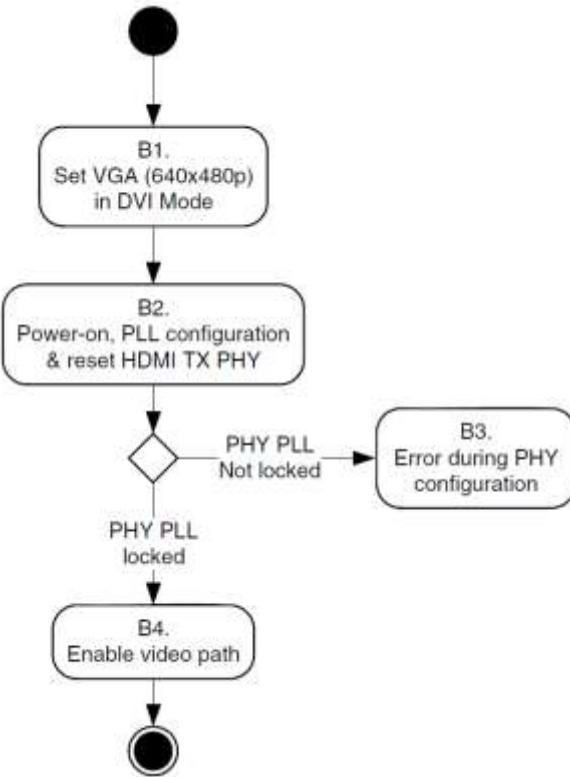


Fig. 5-29 HDMI TX Block Diagram

The VGA (640x480p) mode is mandatory for all receivers and transmitters, because it requires the lowest data rate possible on the HDMI specification.

1. Set VGA (640x480p) in DVI mode.

- To select the Video Mapping input RGB444, 8-bits per color components (24-bits RGB), write “1” in the tx_invid0.video_mapping register.
- To configure VGA timing information (CEA mode 1):
 - Write “0” in the fc_invidconf.vsync_in_polarity register.
 - Write “0” in the fc_invidconf.hsync_in_polarity register.
 - Write “1” in the fc_invidconf.de_in_polarity register.
 - Write “0” in the fc_invidconf.r_v_blank_in_osc register.
 - Write “0” in the fc_invidconf.in_I_P register.
 - 640 H active pixels: 0x280
 - Write “2” in the fc_inhactiv1.H_in_activ register.
 - Write “0x80” in the fc_inhactiv0.H_in_activ register.
 - 480 V active pixels: 0x1E0
 - Write “1” in the fc_invactiv1.V_in_activ register.
 - Write “0xE0” in the fc_invactiv0.V_in_activ register.
 - 160 H blanking pixels: 0xA0
 - Write “0xA0” in the fc_inhblank0.H_in_blank.
 - 45 V blanking pixels: 0x2D
 - Write “0x2D” in the fc_invblank.V_in_blank register.
 - 16 Sync offset: 0x10
 - Write “0x10” in the fc_hsyncindelay0.H_in_delay register.
 - 10 Vsync offset: 0x0A
 - Write “0x0A” in the fc_vsyncindelay0.V_in_delay register.
 - 96 HSync pulse width: 0x60
 - Write “0x60” in the fc_hsyncinwidth0.H_in_width register.
 - 2 VSync pulse width: 0x02

Write “0x02” in the fc_vsyncinwidth0.V_in_width register.

- To select the HDMI mode:

 Write “1” in the fc_invidconf.DVI_modez register.

2. Power-on, configure the PLL, and reset the HDMI TX PHY.

The HDMI TX PHY has an internal PLL that generates TMDS clock from Pixel clock input. The power-on sequence and PLL configuration is part of the HDMI TX PHY documentation.

To configure HDMI TX PHY, you are generally required to:

- Place the PHY in reset by writing 0x01 in the mc_phyrstz register.
- Write the desired color depth and the pixel repetition in the vp_pr_cd register.
- After a PHY-dependent time, it is required to lift the reset by writing 0x00 to the mc_phyrstz register.
- Clear the pddq, txpwron configuration bits, define the data enable polarity and clear the interface control selection by writing these values in the phy_cfg0 register.
- Set the PHY slave address by writing 0x69 in the phy_i2cm_slave register.
- You are required to look up the configuration for your intended video mode and write those values on the PHY I2C interface. The baseline flow to write to the PHY through the I2C interface is:
 - Write register address in the phy_i2cm_address register.
 - Write data in the phy_i2cm_datao_1 (MSB, [15:8]) and phy_i2cm_datao_0 (LSB, [7:0]) registers.
 - Initialize the write operation by writing 8'h10 in the phy_i2cm_operation register.
 - Wait for a done interruption from the I2C master.
- g. After all of the required PHY I2C registers have been configured, you now need to set the txpwron bit in the PHY_CONF0 register, leaving the remaining bits in the state defined previously.

3. Handle errors during PHY configuration.

If the PHY is not correctly configured, the PLL does not lock and no activity can happen on the HDMI output.

4. Enable the video path.

a. Set default parameters in the HDMI TX Controller, Control period duration: 12: 0x0C.

 Write “0x0C” in the fc_ctrldur.ctrlperiodduration bit field register.

b. Set default parameters in the HDMI TX Controller, Extended Control period duration: 32: 0x20.

 Write “0x20” in the fc_exctrldur.exctrlperiodduration bit field register.

c. Set default parameters in the HDMI TX Controller, Max spacing between extended Control period duration: 1: 0x1.

 Write “0x01” in the fc_exctrlspac.exctrlperiodspacing bit field register.

d. Set default parameters in the HDMI TX Controller, Preamble filters to fill TMDS data channels.

- Write “0x0B” in the fc_ch0pream.ch0_preamble_filter bit field register.
- Write “0x16” in the fc_ch1pream.ch1_preamble_filter bit field register.
- Write “0x21” in the fc_ch2pream.ch2_preamble_filter bit field register.

e. Enable pixel clock data path:

 Write “0” in the mc_clkdis.pixelclk_disable bit field register.

f. Enable TMDS clock data path:

- i. Write “0” in the mc_clkdis.tmdsclk_disable bit field register.

- ii. Re-Write the VSync pulse width in the fc_vsyncinwidth0.V_in_width bit field register.

After these steps, it is expected to observe a video picture in VGA mode on the receiver side.

4.6.4 Step C: Reading E-EDID

The E-EDID is a memory present on the receiver side. It contains the video and audio capabilities of the receiver. It is important to read this information and parse it in order to configure the transmitter system in accordance to what the receiver supports. The E-EDID is read using the E-DDC channel present on the HDMI cable.

The E-DDC channel protocol is based on I2C, with segment pointer addressing extension (such as, Extended Read). The hdmi tx controller has an I2C Master controller for this purpose.

1. To perform a “normal” read operation
 - Set I2C slave address.
Write i2cm_slave.slaveaddr[6:0] bit field register.
 - Set I2C register address.
Write i2cm_address.address[7:0] bit field register.
 - Activate Read operation.
Write “1” in the i2cm_operation.rd bit field register.
 - Wait for interruption.
Wait for the ii2cmasterdone interrupt in the ih_i2cm_stat0 register.
 - Read data result.
Read data in the i2cm_datai.datai[7:0] bit field.
2. To perform an E-DDC extended read operation
 - Set I2C slave address.
Write i2cm_slave.slaveaddr[6:0] bit field register.
 - Set I2C segment address.
Write the i2cm_segaddr.seg_addr bit field register.
 - Set I2C segment pointer.
Write i2cm_segptr.segptr bit field register.
 - Activate Read operation.
Write “1” in the i2cm_operation.rd_ext bit field register.
 - Wait for interruption.
Wait for the ii2cmasterdone interrupt in the ih_i2cm_stat0 register.
 - Read data result.
Read data in the i2cm_datai.datai[7:0] bit field register.

4.6.5 Step D: Configure Video Mode

The video mode selected has to match what is supported by the source of the video (Transmitter side) and the sink of the video (receiver side). The video capabilities of the receiver are extracted from the E-EDID information (refer to step C).

To configure any video mode, a similar procedure in step B has to be followed.

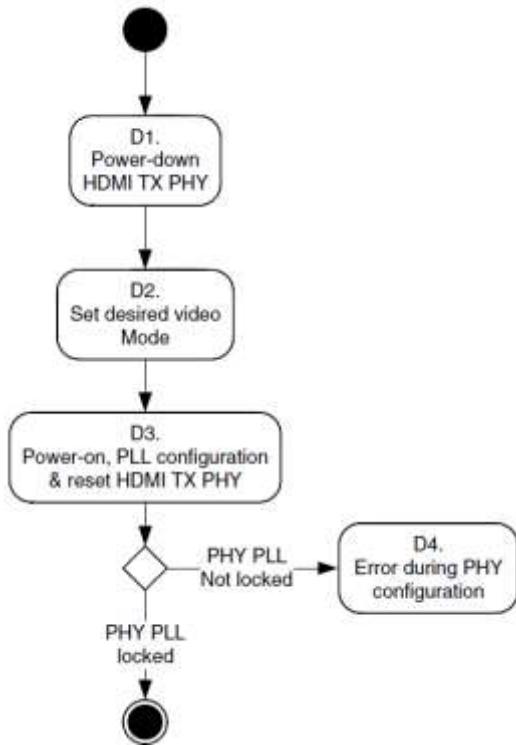


Fig. 5-30 HDMI TX Block Diagram

Before any change is made on the video modes, is it recommended you power-down the HDMI TX PHY to avoid any unexpected behavior on the receiver side.

1. Power-down HDMI TX PHY Detector

- Place the PHY in reset by writing 0x01 in the mc_phyrstz register.
- Then proceed to set SVSRET_MODEZ to 1'b0, PDDQ to 1'b1 and TX_PWRON to 1b0, by writing 2'h12 in the phy_conf0 register.

2. Set desired video mode.

- To select the Video Mapping input mode (RGB444, YCC444, YCC422). Write the video code in the tx_invid0.video_mapping bit field register.
- 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

If the desired video mode is 3D, the fc_invact_2d_1, fc_invact_2d_0, and all registers in the following table must be written.

Table 5-27 HDMI 3D structure table

| 3D STRUCTURE | fc_actspc_hdrl_cfg | |
|--------------------------------------|--------------------|-------------------|
| | fc_actspc_hdrl_tgl | fc_actspc_hdrl_en |
| Frame Packing for interlaced format | 1 | 1 |
| Frame Packing for progressive format | 0 | 1 |
| Field alternative | 0 | 0 |
| Line Alternative | 0 | 0 |
| Side-by-Side(FULL) | 0 | 0 |
| L+depth | 0 | 1 |

| | | |
|---------------------------------|---|---|
| L+depth+graphics+graphics-depth | 0 | 1 |
| Top-and-Bottom | 0 | 0 |
| Reserved for Future Use | 0 | 0 |
| Side-by-Side(FULL) | 0 | 0 |
| Reserved for Future Use | 0 | 0 |
| Not in use | 0 | 0 |

- Write the fc_invact_2D_1 register with the 2D Vertical video active[11:8].

- Write the fc_invact_2D_0 register with the 2D Vertical video active[7:0]

The following registers must always be written with the full V active of the desired video

regardless of the type of the video mode.

- 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.

- Select DVI or HDMI mode:

- Write "1" for HDMI in the fc_invidconf.DVI_modez bit field register.

3. Power-on, configure the PLL, and reset the HDMI TX PHY (same as Step B2).

The HDMI TX PHY has an internal PLL that generates TMDS clock from Pixel clock input. All the power-on sequence and PLL configuration is part of the HDMI TX PHY documentation.

The HDMI TX Controller provides all registers necessary for the PHY sequence implementation, which are presented in the "HDMI PHY Registers".

- Place the PHY in reset by writing 0x01 in the mc_phyrstz register.
- Write in the desired color depth and the pixel repetition in the vp_pr_cd register.
- After a PHY-dependent time, it is required to lift the reset by writing 0x00 to the mc_phyrstz register.
- Clear the pddq, txpwron configuration bits, define the data enable polarity and clear the interface control selection by writing these values in the phy_conf0 register.
- Set the PHY slave address by writing 0x69 to the phy_i2cm_slave register.
- Look up the configuration for your intended video mode and write those values on the PHY I2C interface. The baseline flow to write in the PHY through the I2C interface is
 - Write the register address in the phy_i2cm_address register.
 - Write data into phy_i2cm_datao_1 (MSB, [7:0]) and phy_i2cm_datao_0 (LSB, [7:0]) registers.
 - Initialize the write operation by writing 8'h10 in the phy_i2cm_operation register.
 - Wait for a done interrupt from the I2C master.
- After all of the required PHY I2C registers have been configured, you now need to set the phy_conf0.txpwron register, leaving the remaining bits in the state defined previously.

4.6.6 Step E: 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.

- Configure Audio Input Format with I2S Steps:

- Select I2S input.

Write "1" in the aud_conf0.i2s_select bit field register.

- Enable I2S inputs:

Write "1" in the aud_conf0.i2s_in_en[3:0] bit field register.

- Set I2S Mode [Standard | Right-justified | Left-justified | Burst1 | Burst2]:

Write the aud_conf1.i2s_mode[2:0] bit field register.

- 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:

- Select SPDIF input.

Write "0" in the aud_conf0.i2s_select bit field register.

- Set S/PDIF Linear-PCM or Non-Linear PCM audio samples:

Write the aud_spdif1.setnlpcm bit field register.

- Set SPDIF data width [16 bits up to 24 bits]:

Write the aud_spdif1.spdif_width[4:0] bit field.

- Configure Audio Parameters Steps:

- Set Audio input frequency clock FS ratio factor [128 Fs | 256 Fs | 512 Fs]:

Write the aud_inputclkfs.lfsfactor bit field register.

- 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.

- 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.

- Enable Audio sampler block:

Write "0" in the mc_clkdis.audclk_disable bit field register.

4.6.7 Step F: Configure Infoframes

This step is only relevant when the HDMI TX controller is configured in HDMI mode (refer to "Step D: Configure Video Mode"). In DVI mode, no Infoframes are transmitted. The configuration of the Infoframes is essential to correctly inform the HDMI Receiver about the content of the video and audio format been transmitted. All the detailed information is provided in the HDMI 1.4b Specification.

4.6.8 Step G: Configure HDCP 1.4

Following flow is an initialization of HDCP 1.4. To detect if the HDMI TX Controller is HDCP capable:

1. Read the product_id1.product_id1 bit field register:

- If product_id1 = 01, HDCP is not present.
 - If product_id1 = C1, HDCP is present.
2. Select HDMI mode:
- Write "1" for HDMI in the a_hdcpcfg0.hdmidvi bit field register.
3. Set the Data enable, Hsync and VSync polarity:
Write the dataenpol, vsyncpol, and hsyncpol bit fields of the a_vidpolcfg register.
4. Set encryption on:
- Write "64 (0x40)" for "OessWindowSize" in the a_oesswcfg register.
 - Write "0" for "no HDCP bypass" a_hdcpcfg0.BypassEncryption bit field register.
 - Write "0" for "HDCP enable" in the a_hdcpcfg1.encryptiondisabled bit field register.
5. Reset HDCP engine:
- Write "0" in the a_hdcpcfg1.swreset bit field register.
6. Configure Device Private Keys, which is illustrated in the flow chart Figure 3-6
7. Enable the encryption:
- Write "1" in the a_hdcpcfg0.rxdetect bit field register.
8. When connected to a repeater the SHA-1 calculation is needed to be done by software, which are indicated in the following steps
- a. Wait for an interrupt to be triggered (a_apiintstat.KSVSha1calcint).
 - b. 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).
 - c. Read VH', M0, Bstatus, and the KSV FIFO.
The data is stored in the revocation memory.
 - d. Calculate the SHA-1 checksum (VH) over M0, Bstatus, and the KSV FIFO.
 - e. If the calculated VH equals the VH', set a_ksvmemctrl.SHA1fail to 0 and set a_ksvmemctrl.KSVCTRLupd to 1. If the calculated VH is different from VH' then set a_ksvmemctrl.SHA1fail to 1 and set a_ksvmemctrl.KSVCTRLupd to 1, forcing the controller to re-authenticate from the beginning.

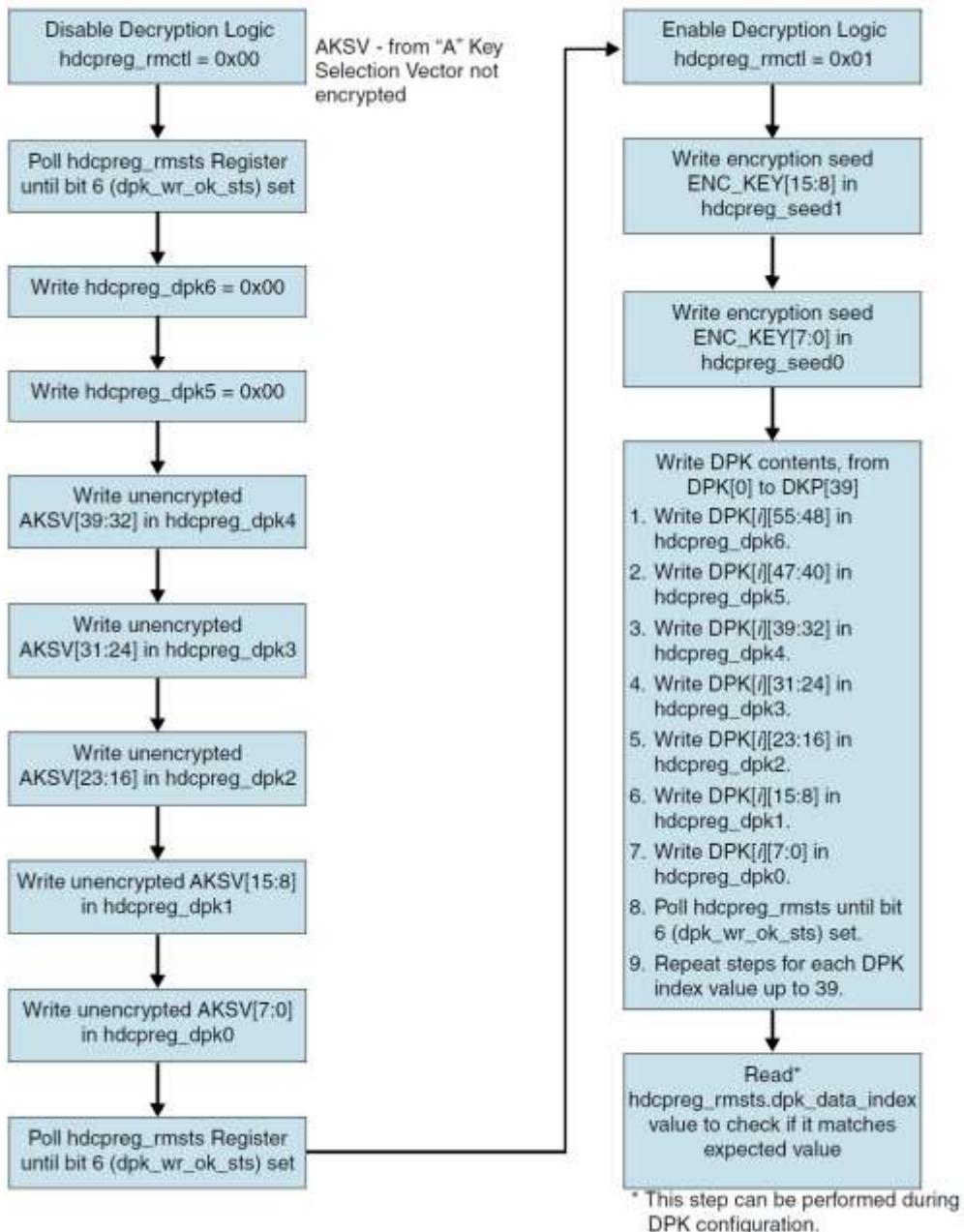


Fig. 5-31 HDMI TX Block Diagram

4.6.9 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:
 - Write the CEC logical address to cec_addr_l, cec_addr_h register
 - Write the size of the frame in bytes which are available in the transmitter data buffer to cec_tx_cnt register
 - Write the desired CEC data(including header and data blocks) to cec_tx_data0 to cec_tx_data15
 - Write 1 to cec_ctrl.send register, to start the cec transmit.

4.6.10 HDMI PHY PLL DESCRIPTION

PRE-PLL OUTPUT CLOCK FREQUENCY EQUATION

The VCO frequency of the pre-PLL($F_{\text{pre-vco}}$) is controlled by pre-pll-pre-divider[4:0] and pre-pll-feedback-divider[8:0]

$$F_{\text{pre-vco}} = (F_{\text{ref}}/\text{pre-pll-pre-divider}[4:0]) * \text{pre-pll-feedback-divider}[8:0]$$

The TMDS clock channel frequency (F_{tx3}) is controlled by tmds-dividera[1:0] and tmds-dividerb[1:0]

$$F_{\text{tx3}} = F_{\text{pre-vco}}/(4 * \text{tmds-dividera}[1:0] * \text{tmds-dividerb}[1:0]),$$

where tmds-dividera[1:0] = 1,2,3,5;

tmds-dividerb[1:0] = 1,2,4,8

The tmdsclk frequency is controlled by tmds-dividera[1:0] and tmds-dividerc[1:0]

$$F_{\text{tmdsclk}} = F_{\text{pre-vco}}/(4 * \text{tmds-dividera}[1:0] * \text{tmds-dividerc}[1:0])$$

where tmds-dividera[1:0] = 1,2,3,5;

tmds-dividerb[1:0] = 1,2,4,8

The frequency of the prepclk equation is

$$F_{\text{prepclk}} = (\text{pclk-dividera}[4:0] == 1)?\{F_{\text{pre-vco}}/(\text{pclk-dividerb}[1:0]*\text{pclk-dividerc}[1:0])\}:\\ \{F_{\text{pre-vco}}/(\text{pclk-dividerb}[4:0]*\text{pclk-dividerc}[1:0])\};$$

where pclk-dividera[4:0] = 1~32;

pclk-dividerb[4:0] = 2,3,4,5;

pclk-dividerc[4:0] = 1,2,4,8;

The frequency of the pclk if $F_{\text{pre-vco}}/5$ when enabling pclk directly divide by 5 from VCO set, otherwise pclk freq is calculated by

$$F_{\text{pclk}} = (\text{pclk-dividera}[4:0] == 1)? \{F_{\text{pre-vco}}/(\text{pclk-dividerb}[1:0]*\text{pclk-dividerc}[1:0])*2\}:\\ \{F_{\text{pre-vco}}/(\text{pclk-dividerb}[4:0]*\text{pclk-dividerc}[1:0])*2\};$$

where pclk-dividera[4:0] = 1~32;

pclk-dividerb[4:0] = 2,3,4,5;

pclk-dividerc[4:0] = 1,2,4,8;

| Resolutions | 8-bit RGB444 480P@60Hz | 8-bit RGB444 1080P@60Hz | 8-bit RGB444 2160P@30Hz | 10-bit YCbCr420 2160P@60Hz |
|-------------------------|---------------------------|----------------------------|----------------------------|-------------------------------|
| pre-divider | 5'd1 | 5'd2 | 5'd2 | 5'd4 |
| feedback-divider | 9'd45 | 9'd99 | 9'd99 | 9'd495 |
| tmds-dividera | 2'b11 | 2'b01 | 2'b00 | 2'b01 |
| tmds-dividerb | 2'b01 | 2'b00 | 2'b00 | 2'b10 |
| tmds-dividerc | 2'b01 | 2'b00 | 2'b00 | 2'b00 |
| pclk-dividera | 5'd1 | 5'd1 | 5'd1 | 5'd1 |
| pclk-dividerb | 2'b11 | 2'b10 | 2'b00 | 2'b11 |
| pclk-dividerc | 2'b11 | 2'b01 | 2'b01 | 2'b01 |
| pclk-dividerd | 5'd4 | 5'd1 | 5'd1 | 5'd1 |
| enable divided from VCO | 1'b0 | 1'b0 | 1'b0 | 1'b1 |

Fig. 5-32 Pre-PLL Configuration for Main Typical Resolution

POST-PLL OUTPUT CLOCK FREQUENCY EQUATION

The post-pll just only used to generate differential serial clock for drivers which differential frequency ($F_{\text{diff-sclk}}$) is always 5 times of the tmdsclk.

$$F_{\text{post-vco}} = (F_{\text{tmdsclk}}/\text{post-pll-pre-divider}[4:0]) * \text{post-pll-feedback-divider}[8:0]$$

The $F_{\text{diff-sclk}}$ equal $F_{\text{post-vco}}$ when the post-pll post divider enable is clear (2'b00), otherwise.

$$F_{\text{diff-sclk}} = F_{\text{post-vco}}/\text{post-pll-post-divier}[1:0]$$

where post-pll-post-divider = 2,4,8

Chapter 5 HDCP2.2 Controller

5.1 Overview

The HDCP22 controller is an Embedded Security Module (ESM) inside HDMI TX controller. It is a self-contained module pre-integrated with the HDMI controller to ensure DCP robustness rules for HDCP. The ESM has a small MCU inside. It reads ESM Image for operation through the AXI interface.

The HDCP22 Controller supports following features:

- Supports HDCP Revision 2.2
- Bus Interface Features:
 - Supports AXI interface with MMU
 - Supports AHB interface for MMU configuration
 - Supports APB interface for processor to send commands and get status from the ESM (Host Port Interface, HPI)
 - A Secure Key Port (SKP) for cryptographic keys and random nonce
- MMU:
 - 4k/64k page size
 - TLB pre-fetch

5.2 Block Diagram

The Host Controller consists of the following main functional blocks.

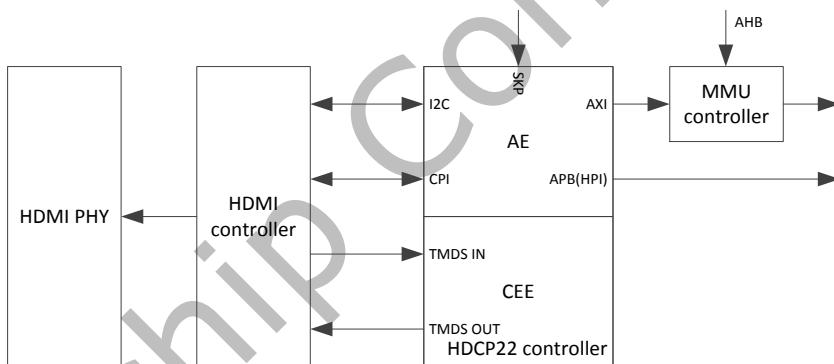


Fig. 5-1 hdcp22 Controller Block Diagram

The embedded security module is composed of two parts:

- The Authentication Engine (AE) contains a controller that runs the HDCP authentication process.
- The Content Encryption Engine (CEE) is an ESM component that encrypts data for the HDCP transmitter.

5.3 Function Description

5.3.1 AXI interface

The ESM requires continuous access to its ESM Image for operation, which is accessed through the AXI interface and usually placed a reserved portion of DDR memory. It also uses this interface for communicating messages with the external host processor to pass content such as SRM data, pairing information, and debug/logging information.

The clock used by ESM requires a minimum frequency: minimum ratio of 1:2.3 (hdcp axi clock: VOP Pixel clock) for proper operation. For an hdmi output display 4K resolution at 600 MHz, this clock can be no less than 261 MHz.

Following Table summarizes the various requirements the ESM places on the RK3228A/RK3228B's memory, assuming an AXI interface clock of 261 MHz and an average latency of 1000 ns.

Table 5-1 ESM average memory bandwidth usage

| Phase | Duration (ms) | Appropriate Bandwidth (Mbytes/sec) | Description |
|---------------------|---------------|------------------------------------|---|
| Bootup | 50 | 5.0 | Initial startup of ESM to Idle state. |
| Tx Authentication | 525ms | 1.5 | ESM transmitter authentication phase including capability check. |
| Idle, authenticated | n/a | < 1.0, 4.0 | ESM is either in the Idle state waiting to start authentication or in the authenticated state. The second number is when the ESM has debug logging enabled. |

The external host memory required by the ESM must be physically contiguous and allocated before the ESM can be started. The ESM Image and R/W memory buffer addresses must be 4K aligned otherwise the ESM cannot properly access the memory. Following table describes the requirements.

Table 5-2 ESM average memory bandwidth usage

| Size (bytes) | ESM Access | Purpose |
|--|--------------|--|
| ~160K | Read Only | Contains the ESM Image. Note this size is approximate and varies depending on the image type (Rx/Tx) and the current version of the ESM Image. |
| 16K + n*256 (Tx) (n is the number of pairs supported) | Read / Write | Bidirectional communication buffer to exchange data with the Host. Messages such as SRM updates, pairing data, and logging information are placed in this area by the host and the ESM. The number of pairs supported by this parameter is defined in the Tx configuration file ([MEM_PAIRING_SIZE]). |

In RK3228A/RK3228B the provided ESM host application library allocates a small amount of the processor's memory space, which is accessible by the ESM. Communication with this interface is provided by the ESM host application library.

5.3.2 AHB interface

MMU register is accessed from AHB interface.

5.3.3 APB interface (HPI)

The external host issues commands to the ESM through the HPI. The ESM supports a limited set of commands such as:

- Enabling and disabling of security on the link
- Enabling debugging/logging
- Retrieving pairing information
- Updating the SRM
- ESM status

The external host must issue a SRM list to the ESM when one is available, according to the requirements of DCP LLC.

Communication with this interface is provided by the ESM host application library.

5.3.4 SKP Interface

The SKP is a secure interface that contains the secret keys for the ESM along with an input for a True Random Number Generator (TRNG). The keys are used by the ESM to create session keys and other cryptographic secrets required for operation. Customers should put following keys in EFUSE for secure reason.

Table 5-3 Secure Key for ESM

| Description | Purpose |
|-------------------------|--|
| Platform Key (PKf) | This key is used to decrypt the ESM Image. The PKf is intended to be common across a deployed model of a product, which allows field upgrades of the ESM Image should it be necessary. |
| Device Unique Key (DUK) | This key is used to decrypt the DCP Rx key data and to encrypt and decrypt content unique to the ESM such as the pairing message content. This value should be unique for every unit. |

Note: The PKf and DUK must be kept confidential as exposing either one compromises the security of the ESM.

5.3.5 HDCP22 Controller Behavior

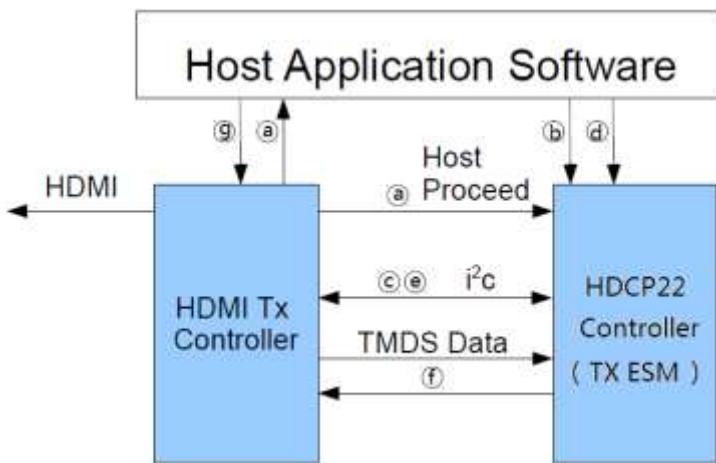


Fig. 5-2 Transmitter Authentication

Hdcp22 controller in RK3228A/RK3228B is a transmitter, it works as following steps:

- The controller asserts Host Proceed to the ESM. The application software receives the notification from the controller.
- The application software requests the ESM to perform a capability check with the receiver.
- The ESM queries the receiver. If the receiver does not support HDCP 2.2 then the ESM idles and the application software asks the HDMI controller to check if the receiver supports HDCP 1.4 (or no encryption) and takes appropriate action. The ESM is no longer required.
- If the receiver supports HDCP 2.2, the application software requests the ESM initiate authentication.
- The ESM initiates authentication with the receiver and successfully completes authentication (or not). The application software and the HDMI controller are appropriately notified.
- If authentication is successful the ESM starts to encrypt the TMDS data.
- The application software can now send high value content securely on the HDMI link.

5.3.6 HDCP22 Software Initialization

An external host controller (ARM Core or MCU) allocates memory, initializes the ESM, and enables the ESM controller. The general reset and startup sequence is:

- Configure clocks for hdcp controller.
- Load PKF & DUK from effuse to sgrf.
- The ESM reset is deasserted and the ESM enters a reset state.
- The host allocates contiguous memory for the ESM to use. The ESM Image and R/W area buffers must be 4K aligned.
- The host configures the registers of the ESM with the physical memory pointers for the ESM image and communications buffer.
- The host copies the ESM Image from a persistent storage location (or other memory region) to the memory.
- The host configures itself to support interrupts from the APB Interface (HPI) of the ESM for processing commands.

- h. The host enables the ESM controller to a running state. At this point the ESM initiates transactions on the AXI bus to fetch its Image.
- i. ESM reports a ready status to the host when it can accept commands.

The ESM host application library provides the startup sequence from steps d through i.

5.3.7 Key Usage

The ESM uses the DUK to protect specific content such as pairing data. When requesting pairing data from the ESM, the information is encrypted with the unique DUK thus preventing its use with any other ESM—the information is unique to the issuing unit, preventing the key pairing data from being used by another one.

The ESM uses the PKf to decrypt the Image it uses for operation. The ESM Image must be processed with tools that are provided with the ESM software package. Following figure illustrates the tools and the flow that are required to produce a deployable ESM Image.

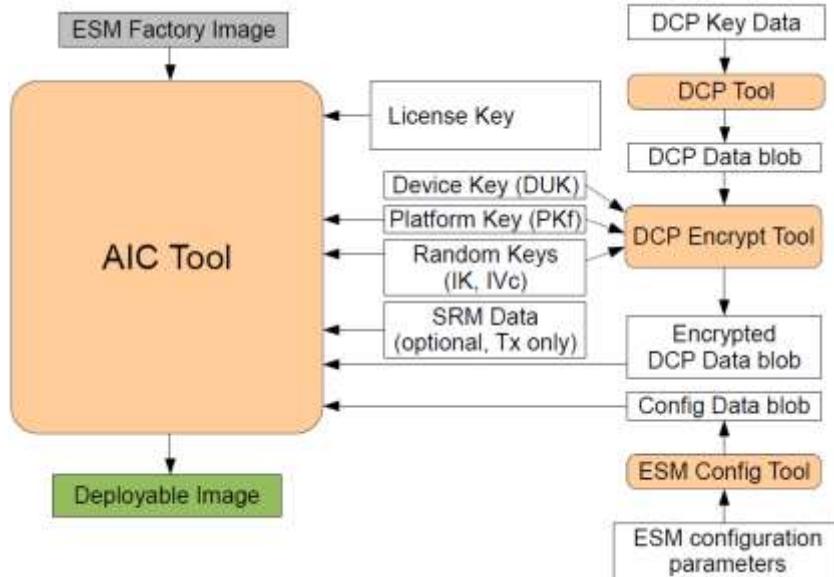


Fig. 5-3 ESM Image tool flow

For a RK3228A/RK3228B, the DCP key data is common. Note that while the image has an option where a default SRM list can be optionally inserted as part of the manufacturing process, the SRM update can also be sent to the ESM as a separate command after the ESM has been booted.

In some cases it may be necessary to provide a field update of the ESM Image after deployment, either because of a factory update or a change to the ESM configuration file. Updated image can be issued using the process noted above as all field Tx units receive the same DCP key.

5.4 Register Description

The registers listed following are used for MMU interface. Any operation to hdcp22 controller is done through Run Time Library(See next chapter: Application Notes).

5.4.1 Register Summary

| Name | Offset | Size | Reset Value | Description |
|----------------------------|---------|------|-------------|--|
| HDCP22_MMU_DTE_ADDR | 0x00000 | W | 0x00000000 | MMU current page Table address |
| HDCP22_MMU_STATUS | 0x00004 | W | 0x00000018 | MMU status register |
| HDCP22_MMU_COMMAND | 0x00008 | W | 0x00000000 | MMU command register |
| HDCP22_MMU_PAGE_FAULT_ADDR | 0x0000c | W | 0x00000000 | MMU logical address of last page fault |
| HDCP22_MMU_ZAP_ONE_LINE | 0x00010 | W | 0x00000000 | MMU Zap cache line register |

| Name | Offset | Size | Reset Value | Description |
|--------------------------|---------|------|-------------|-----------------------------------|
| HDCP22_MMU_INT_RAWSTAT | 0x00014 | W | 0x00000000 | MMU raw interrupt status register |
| HDCP22_MMU_INT_CLEAR | 0x00018 | W | 0x00000000 | MMU raw interrupt status register |
| HDCP22_MMU_INT_MASK | 0x0001c | W | 0x00000000 | MMU raw interrupt status register |
| HDCP22_MMU_INT_STATUS | 0x00020 | W | 0x00000000 | MMU raw interrupt status register |
| HDCP22_MMU_AUTO_GATING | 0x00024 | W | 0x00000001 | mmu auto gating |
| HDCP22_MMU_mmu_miss_cnt | 0x00028 | W | 0x00000000 | Register0000 Abstract |
| HDCP22_MMU_mmu_burst_cnt | 0x0002c | W | 0x00000000 | Register0001 Abstract |

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

5.4.2 Detail Register Description

HDCP22_MMU_DTE_ADDR

Address: Operational Base + offset (0x00000)

MMU current page Table address

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:0 | RW | 0x00000000 | MMU_DTE_ADDR mmu dte base addr mmu dte base addr , the address must be 4kb aligned |

HDCP22_MMU_STATUS

Address: Operational Base + offset (0x00004)

MMU status register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 31:11 | RO | 0x0 | reserved |
| 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 | 0x1 | REPLAY_BUFFER_EMPTY Field0000 Abstract The MMU replay buffer is empty |
| 3 | RO | 0x1 | MMU_IDLE Field0000 Abstract The MMU is idle when accesses are being translated and there are no unfinished translated accesses. |

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

HDCP22_MMU_COMMAND

Address: Operational Base + offset (0x00008)

MMU command register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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_PAGE_FAULT_DONE 6: MMU_FORCE_RESET |

HDCP22_MMU_PAGE_FAULT_ADDR

Address: Operational Base + offset (0x0000c)

MMU logical address of last page fault

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RO | 0x00000000 | PAGE_FAULT_ADDR Field0000 Abstract address of last page fault |

HDCP22_MMU_ZAP_ONE_LINE

Address: Operational Base + offset (0x00010)

MMU Zap cache line register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | WO | 0x00000000 | MMU_ZAP_ONE_LINE Field0000 Abstract address to be invalidated from the page table cache |

HDCP22_MMU_INT_RAWSTAT

Address: Operational Base + offset (0x00014)

MMU raw interrupt status register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:2 | RO | 0x0 | reserved |
| 1 | RW | 0x0 | READ_BUS_ERROR Field0000 Abstract read bus error |
| 0 | RW | 0x0 | PAGEFAULT Field0000 Abstract page fault |

HDCP22_MMU_INT_CLEAR

Address: Operational Base + offset (0x00018)

MMU raw interrupt status register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:2 | RO | 0x0 | reserved |
| 1 | WO | 0x0 | READ_BUS_ERROR Field0000 Abstract read bus error |
| 0 | WO | 0x0 | PAGEFAULT Field0000 Abstract page fault |

HDCP22_MMU_INT_MASK

Address: Operational Base + offset (0x0001c)

MMU raw interrupt status register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:2 | RO | 0x0 | reserved |
| 1 | RW | 0x0 | READ_BUS_ERROR Field0000 Abstract read bus error enable an interrupt source if the corresponding mask bit is set to 1 |
| 0 | RW | 0x0 | PAGEFAULT Field0000 Abstract page fault enable an interrupt source if the corresponding mask bit is set to 1 |

HDCP22_MMU_INT_STATUS

Address: Operational Base + offset (0x00020)

MMU raw interrupt status register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:2 | RO | 0x0 | reserved |
| 1 | RO | 0x0 | READ_BUS_ERROR Field0000 Abstract read bus error |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 0 | RO | 0x0 | PAGE_FAULT Field0000 Abstract page fault |

HDCP22_MMU_AUTO_GATING

Address: Operational Base + offset (0x00024)

mmu auto gating

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:1 | RO | 0x0 | reserved |
| 0 | RW | 0x1 | mmu_auto_gating mmu auto gating when it is 1'b1, the mmu will auto gating it self |

HDCP22_MMU_mmu_miss_cnt

Address: Operational Base + offset (0x00028)

Register0000 Abstract

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31 | RW | 0x0 | cnt_ctrl_sel sel the counter for mmu_miss or mmu_real_miss 1'b0: mmu_real_miss 1'b1: mmu_miss When sel 1'b1, an axi command miss may count for several times; when sel 1'b0, an axi command only count for a time |
| 30 | RW | 0x0 | miss_cnt_overflow_flag miss cnt overflow flag miss cnt overflow flag |
| 29:0 | RW | 0x00000000 | miss_cnt count for miss AXI command count for miss AXI command |

HDCP22_MMU_mmu_burst_cnt

Address: Operational Base + offset (0x0002c)

Register0001 Abstract

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31 | RO | 0x0 | reserved |
| 30 | RW | 0x0 | burst_cnt_overflow_flag burst cnt overflow flag burst cnt overflow flag |
| 29:0 | RW | 0x00000000 | burst_cnt The AXI input burst counter The AXI input burst counter |

5.5 Application Notes

5.5.1 Host Application Software

The ESM package contains a series of tools and a run-time library as below:

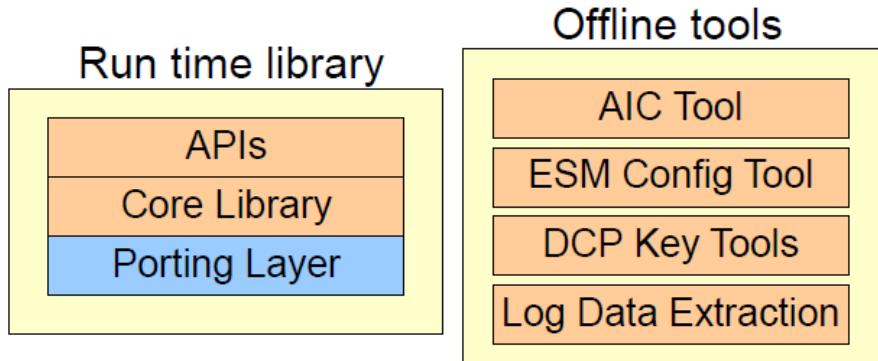


Fig. 5-4 ESM software

5.5.2 Run Time Library

This library is provided as source code and is used by the host application processor to control the ESM.

Setting the configurable options of the ESM is not provided through APIs for security reasons. It is instead embedded into the ESM's Image as part of the process of building a deployable image.

The APIs provide a minimal interface to the ESM, which include the following:

- Load/Initialize firmware image
- Start authentication to secure a link
- Terminate an authenticated link
- Collect logging data
- Get/Set pairing information
- Update SRM data
- Status updates

Sample code is provided and demonstrates how to use the various APIs. In addition, a software API reference guide is included and provides details on the API calls and their parameters.

5.5.3 Offline Tools

There are three separate offline tools that are required to build a deployable image for the ESM as shown in Figure ESM Image tool flow and one tool for logging information. All of these tools are developed to run on a 32 bit Linux system.

5.5.4 ESM Config Tool

This tool is used to insert configuration parameters into the ESM Image before deployment, including debugging parameters, which can be set to assist in-house development and debugging. These parameters cannot be changed in the field with software API calls for security reasons, and any debugging parameters must be disabled before production deployment as they can be used to disable certain security features.

This tool reads a configuration file containing parameters. The purpose of the parameter is documented in the commented sections. A sample set of user adjustable parameters are:

- Disable timeouts (debug only)
- Bypass (debug only)
- Pairing limit
- Enable logging and set logging levels (min to max)

Sample configuration files are located at samples/tx_config.txt

The following sample command lines are used to generate TX configuration.

.esm_tx_config -i ..samples/tx_config.txt -o data_tx.cfg

Where the data_tx.cfg are the formatted configuration files that must be passed to the AIC

tool to generate the final image.

There is no need to run this tool separately. A makefile is provided that runs all three tools in sequence to create the proper output.

ESM Configuration File

The [CPU_FREQ] parameter in the configuration file must be changed to permit the ESM to operate properly in your system. This value must be set to the frequency of the AXI clock input to the ESM.

The [CEE_BSOD] is a 32 bit decimal value which parameter represents fixed data output on the TMDS data bus if the ESM encounters an authentication issue or detects attempts have been made to tamper with it. This 32 bit value is output as follows:

Table 5-4 ESM BSOD Output Mapping

| BSOD Bit Positions | HDMI Controller TMDS Data Channel |
|---------------------------|--|
| 7..0 | ch0[7..0] |
| 15..8 | ch1[7..0] |
| 23..16 | ch2[7..0] |
| 32..24 | Not used |

5.5.5 DCP Key Tools

These tools are used to create a DCP key data file, which is required to build the encrypted DCP data blob needed for building the final ESM Image. One tool is used to create the obfuscated data blob and another tool is used to encrypt the DCP key blob before its inclusion into the final ESM Image. (This tool also reads the configuration file firmware.aic to obtain the secret key values).

A sample file containing the DCP test keys is provided for reference.

The information shown in following table is required:

Table 5-5 Requirements for DCP Tools

| Confidential Elements | Tag | Size |
|--------------------------------|----------------|-------------|
| lc128 | [LC_128] | 16 |
| 3072 RSA DCP Public Key (base) | [KPUB_DCP_MOD] | 384 |
| 3072 RSA DCP Public Key (exp) | [KPUB_DCP_E] | 3 |

5.5.6 AIC Tool

This tool is used to create a deployable ESM Image. To build an ESM image, you must first create the data file for the DCP keys and the configuration. These files, along with an input configuration file for this tool are used to build an image that can be used by the ESM. In addition to specifying the Platform and Device Unique Keys in the configuration file, there are additional keys that also must be specified as noted below.

Content Randomization

There are two separate values, IK and IVc (128 bits and 96 bits respectively), which must be randomly generated for each creation of a deployed ESM Image. It is mandatory that any time an ESM Image is generated to be field deployed that new random values for each are created and used. Failure to do this puts secret data in the ESM Image at risk of being compromised.

Log Data Extraction

The logging information from the ESM is output in a proprietary format. This tool is used to produce readable output of captured logging data content.

The AICTOOL program is used to assemble the following files:

- ESM Factory Image file (.rom)
- Configuration file (*.cfg)
- HDCP KEY file (*.out).

The tool takes these three files and builds an encrypted image that can be used by an ESM (the ESM Image). Output ROM file located at bin/firmware_tx.rom

This tool reads an input configuration file (firmware/firmware.aic) to obtain the key values required to build the encrypted ESM Image. Table below describes the user adjustable parameters in this file. Any parameter not documented must not be modified.

Table 5-6 Description of user adjustable parameters

| Parameter | Description |
|------------------|---|
| PKf | The 128 bit value present on the PKf input of the SKP |

| | |
|---------|---|
| DUK | The 128 bit value present on the DUK input of the SKP |
| IK, IVC | Content randomization values as described in this section |

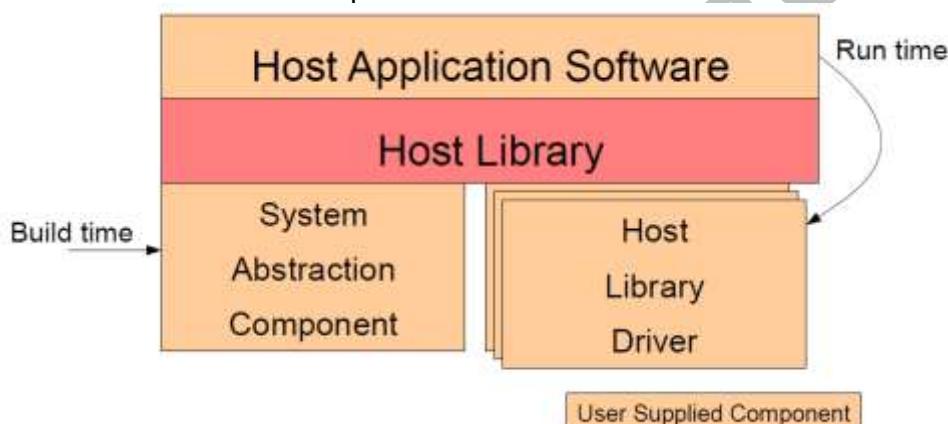
5.5.7 Sample Application Software

The Host Library contains a core library component which includes all the functions necessary to use the ESM. The library is built using portable C code and uses various abstraction functions to allow for migration to different platforms. Before using the ESM in your system you must properly implement the abstraction components specific to your platform.

There are two distinct abstraction components used by the Host Library, the System Abstraction Layer and the Host Library Driver (HLD) component. The System Abstraction Layer (or Common Component) defines system level functions used by the Host Library, such as malloc, memcpy, etc. These functions are linked into the Host Library when it is built.

The Host Library Driver (HLD) component establishes a communication between the library and the physical ESM hardware, including managing the memory required for the ESM Image and the read/write area. This is a run-time plug-in module that is defined in your application and specified as a parameter to the ESM initialization function.

The figure below illustrates the abstraction components used by the Host Library package. This figure demonstrates one example where there are three separate ESMs in the system, requiring (typically) three separate HLD instantiations. Consult the ESM Host Library API guide for more details on the HLD implementation.



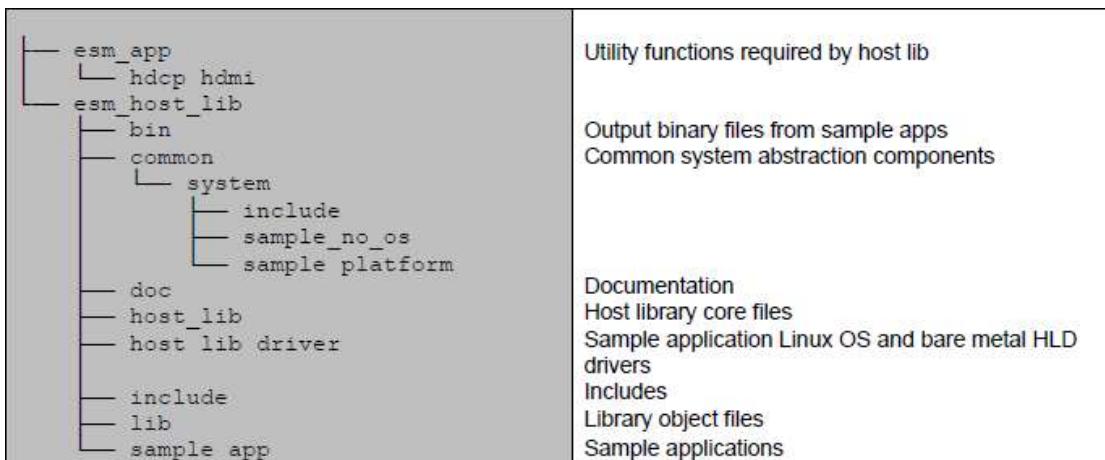


Fig. 5-6 The development tree overview

There are two sample transmitter applications provided in the sample_app folder:
 hdcp_tx.c is a Linux based example and utilizes file IO and threads.
 hdcp_tx_bm.c is a bare metal example.

Execution Environments

The provided development environments come with two sample platform targets: Linux OS and a bare metal platform as shown in the following table.

Table 5-7 Common Platform Target Examples

| Common Samples Provided | Description |
|-------------------------|--|
| sample_platform | Demonstrates the system API implementations when targeting a Linux OS platform |
| sample_no_os | Demonstrates the system API implementations when targeting a platform with no operating system (also known as bare-metal). |

Linux OS

These sample applications are targeted for a Linux User Space application environment. In order to use them a custom kernel driver was created and the HLD plug-in for the host library was made to call IOCTLs which mapped to the appropriate interfaces in the kernel driver module to access the ESM. To build for this platform run the makefile from the esm_host_lib folder:

make samples

The implementation of the platform specific APIs (for the Linux example) are in the following files:

./common/system/sample_platform/elliptic_common.c
 ./common/system/sample_platform/elliptic_log.c
 ./common/system/sample_platform/include/elliptic_platform_specific_system_type.h
 ./common/system/sample_platform/include/elliptic_platform_specific_system.h

You can change these implementations as needed for your specific platform.

No OS (Bare Metal)

This sample demonstrates how to build an application for a no OS (bare metal) environment with direct access to memory and system functions. In the supplied example there are no implementations for the HLD and you must first create them appropriate to your system. When you run the make example above it also build this reference application as well.

The implementation of the platform specific APIs are in the following files:

./common/system/sample_no_os/elliptic_common.c
 ./common/system/sample_no_os/elliptic_log.c
 ./common/system/sample_no_os/include/elliptic_platform_specific_system_type.h
 ./common/system/sample_no_os/include/elliptic_platform_specific_system.h

There are currently no implementations and you must create them as appropriate to your platform.

Rockchip Confidential

Chapter 6 Pulse Width Modulation (PWM)

6.1 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
 - The capture result of channel 3 can be stored in a FIFO. The depths of FIFO is 8, and the data in FIFO can be read through DMA. It also supports timeout interrupt when the data in FIFO has not been read in a time threshold.
 - 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 does not generate 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.

6.2 Block Diagram

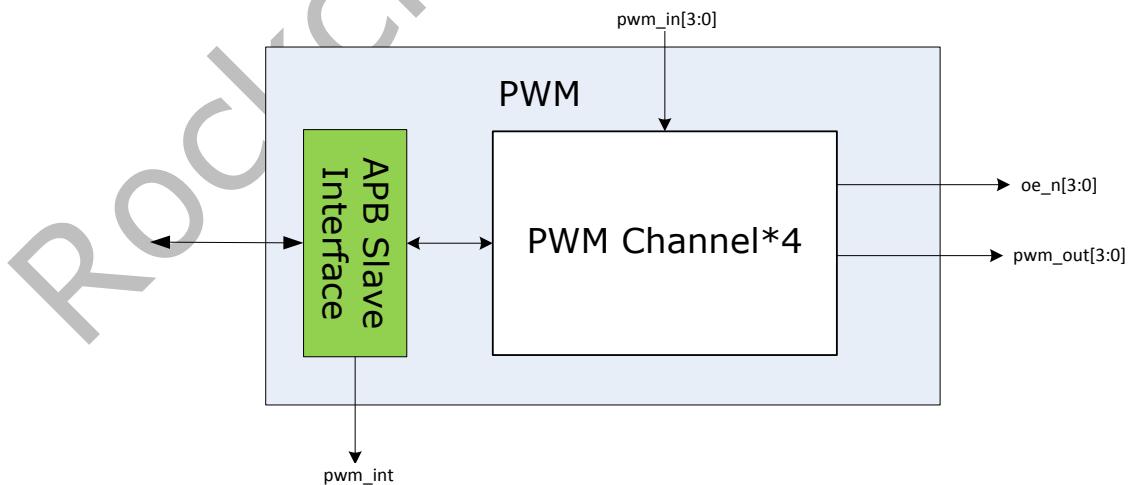


Fig. 5-7 PWM Block Diagram

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. PWM only supports one interrupt output, please refer to interrupt register to know the raw interrupt status when an interrupt is asserted.

PWM Channel is the control logic of PWM module, and controls the operation of PWM module.

according to the configured working mode.

6.3 Function Description

The PWM supports three operation modes: capture 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.

6.3.1 Capture mode

The capture 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.

Notes: 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.

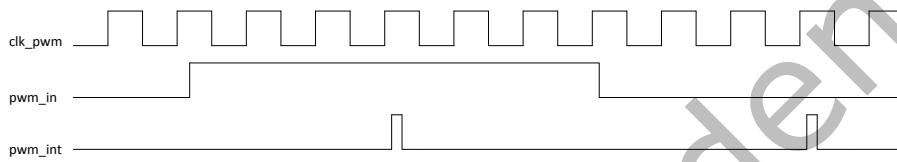


Fig. 5-8 PWM Capture Mode

6.3.2 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.

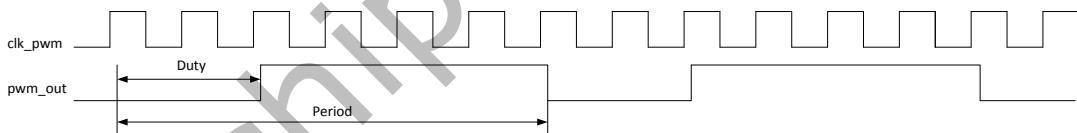


Fig. 5-9 PWM Continuous Left-aligned Output Mode

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.

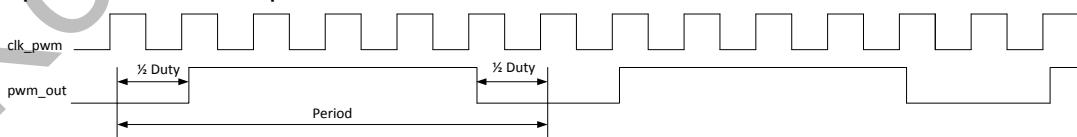


Fig. 5-10 PWM Continuous Center-aligned Output Mode

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).

6.3.3 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.

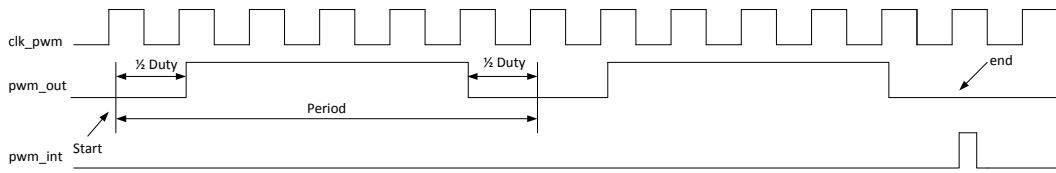


Fig. 5-11 PWM One-shot Center-aligned Output Mode

6.4 Register Description

6.4.1 Registers Summary

| 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 |
| PWM_PWM_FIFO_CTRL | 0x0050 | W | 0x00000000 | PWM Channel 3 FIFO Mode Control Register |
| PWM_PWM_FIFO_INTSTS | 0x0054 | W | 0x00000000 | FIFO Interrupts Status Register |
| PWM_PWM_FIFO_TOUTTH_R | 0x0058 | W | 0x00000000 | FIFO Timeout Threshold Register |
| PWM_PWM_FIFO | 0x0060 | W | 0x00000000 | FIFO Register |

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

6.4.2 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_HPR 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 |
|------|------|-------------|---|
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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 field 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 |
| 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 |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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 |
| 0 | RW | 0x0 | pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked in 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

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:0 | RW | 0x00000000 | PERIOD_HPR 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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 field 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^{10}). |
| 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 |
| 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 |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 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 |
| 0 | RW | 0x0 | pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked in 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

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

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:0 | RW | 0x00000000 | PERIOD_HPR 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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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 field 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 |
| 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 |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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 |
| 0 | RW | 0x0 | pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked in 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

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:0 | RW | 0x00000000 | PERIOD_HPR 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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 field 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 |
| 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 |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 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 |
| 0 | RW | 0x0 | pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked in the one-shot mode, this bit will be cleared at the end of operation |

PWM_INTSTS

Address: Operational Base + offset (0x0040)

Interrupt Status Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|-------------|
| 31:12 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 11 | RO | 0x0 | <p>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 3 input waveform. Otherwise, please refer to PWM3_PERIOD_LPR to know the effective low cycle of Channel 3 input waveform. Write 1 to CH3_IntSts will clear this bit.</p> |
| 10 | RO | 0x0 | <p>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 2 input waveform. Otherwise, please refer to PWM2_PERIOD_LPR to know the effective low cycle of Channel 2 input waveform. Write 1 to CH2_IntSts will clear this bit.</p> |
| 9 | RO | 0x0 | <p>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 1 input waveform. Otherwise, please refer to PWM1_PERIOD_LPR to know the effective low cycle of Channel 1 input waveform. Write 1 to CH1_IntSts will clear this bit.</p> |
| 8 | RO | 0x0 | <p>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.</p> |
| 7:4 | RO | 0x0 | reserved |
| 3 | RW | 0x0 | <p>CH3_IntSts Channel 3 Interrupt Status 0: Channel 3 Interrupt not generated 1: Channel 3 Interrupt generated</p> |
| 2 | RW | 0x0 | <p>CH2_IntSts Channel 2 Interrupt Status 0: Channel 2 Interrupt not generated 1: Channel 2 Interrupt generated</p> |
| 1 | RW | 0x0 | <p>CH1_IntSts Channel 1 Interrupt Status 0: Channel 1 Interrupt not generated 1: Channel 1 Interrupt generated</p> |

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

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

PWM_PWM_FIFO_CTRL

Address: Operational Base + offset (0x0050)

PWM Channel 3 FIFO Mode Control Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:10 | RO | 0x0 | reserved |
| 9 | RW | 0x0 | timeout_en FIFO Timeout Enable |
| 8 | RW | 0x0 | dma_mode_en DMA Mode Enable 1'b1: enable 1'b0: disable |
| 7 | RO | 0x0 | reserved |
| 6:4 | RW | 0x0 | almost_full_watermark Almost Full Watermark Level |
| 3 | RW | 0x0 | watermark_int_en Watermark Full Interrupt |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 2 | RW | 0x0 | overflow_int_en FIFO Overflow Interrupt Enable When high, an interrupt asserts when the channel 3 FIFO is overflow. |
| 1 | RW | 0x0 | full_int_en FIFO Full Interrupt Enable When high, an interrupt asserts when the channel 3 FIFO is full. |
| 0 | RW | 0x0 | fifo_mode_sel FIFO MODE Sel When high, PWM FIFO mode is activated |

PWM_PWM_FIFO_INTSTS

Address: Operational Base + offset (0x0054)

FIFO Interrupts Status Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:5 | RO | 0x0 | reserved |
| 4 | RO | 0x0 | fifo_empty_status FIFO Empty Status This bit indicates the FIFO is empty |
| 3 | W1C | 0x0 | timeout_intsts Timeout Interrupt |
| 2 | W1C | 0x0 | fifo_watermark_full_intsts FIFO Watermark Full Interrupt Status This bit indicates the FIFO is Watermark Full |
| 1 | W1C | 0x0 | fifo_overflow_intsts FIFO Overflow Interrupt Status This bit indicates the FIFO is overflow |
| 0 | W1C | 0x0 | fifo_full_intsts FIFO Full Interrupt Status This bit indicates the FIFO is full |

PWM_PWM_FIFO_TOUTTHR

Address: Operational Base + offset (0x0058)

FIFO Timeout Threshold Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:20 | RO | 0x0 | reserved |
| 19:0 | RO | 0x00000 | timeout_threshold FIFO Timeout Value(unit pwmclk) |

PWM_PWM_FIFO

Address: Operational Base + offset (0x0060)

FIFO Register

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31 | RW | 0x0 | pol Polarity This bit indicates the polarity of the lower 31-bit counter. 0: Low 1: High |
| 30:0 | RO | 0x00000000 | cycle_cnt High/Low Cycle Counter This 31-bit counter indicates the effective cycles of high/low waveform. |

6.5 Interface Description

Table 5-8 PWM Interface Description

| Module Pin | Direction | Pad Name | IOMUX Setting |
|------------|-----------|---------------------------------|--|
| PWM0 | I/O | IO_PWM0_PCMRx _GPIO0d2 | GRF_GPIO0D_IOMUX[5:4]=2'b01 GRF_CON_IOMUX[0]=1'b0 |
| | | IO_PWM10_GPIO3c5 | GRF_GPIO3C_IOMUX[11:10]=2'b01 GRF_CON_IOMUX[0]=1'b1 |
| PWM1 | I/O | IO_PWM1_PCMTx _GPIO0d3 | GRF_GPIO0D_IOMUX[7:6]=2'b01 GRF_CON_IOMUX[1]=1'b0 |
| | | IO_SDIOpwren _PWM11_GPIO0d6 | GRF_GPIO0D_IOMUX[13:12]=2'b10 GRF_CON_IOMUX[1]=1'b1 |
| PWM2 | I/O | IO_PWM2_GPIO0d4 | GRF_GPIO0D_IOMUX[9:8]=2'b01 GRF_CON_IOMUX[2]=1'b0 |
| | | IO_SPIcsn1_PWM12 _GPIO1b4 | GRF_GPIO1B_IOMUX[9:8]=2'b10 GRF_CON_IOMUX[2]=1'b1 |
| PWM3 | I/O | IO_PWMir_GPIO3d2 | GRF_GPIO3D_IOMUX[5:4]=2'b01 GRF_CON_IOMUX[3]=1'b0 |
| | | IO_UART1rtsn _PWM1ir_GPIO1b3 | GRF_GPIO1B_IOMUX[7:6]=2'b10 GRF_CON_IOMUX[3]=1'b1 |

Notes: I=input, O=output, I/O=input/output, bidirectional.

Application Notes

6.5.1 PWM Capture 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 capture 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.

6.5.2 PWM Capture DMA 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 3 to work in the capture mode.
4. Configure the PWM_FIFO_CTRL.dma_mode_en and PWM_FIFO_CTRL fifo_mode_sel to enable the DMA mode. Configure PWM_FIFO_CTRL.almost_full_watermark at appropriate value.
5. Configure DMAC to transfer data from PWM to DDR.
6. Enable the channel by writing '1' to PWMx_CTRL.pwm_en bit to start the channel.
7. When an dma_req is asserted, DMAC transfer the data of effective high cycles and low cycles of input waveforms to DDR.
8. Write '0' to PWMx_CTRL.pwm_en to disable the channel.

6.5.3 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 in, write '0' to PWMx_CTRL.pwm_en bit to disable the PWM channel.

6.5.4 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.

6.5.5 Other notes

When the channel is active to produce waveforms, it is free to program 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.

Chapter 7 UART

7.1 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:

- Support 3 independent UART controller: UART0, UART1, UART2
- UART0/UART1/UART2 all contain two 64Bytes FIFOs for data receive and transmit
- UART0/UART1/UART2 all support auto flow-control
- Support bit rates 115.2Kbps, 460.8Kbps, 921.6Kbps, 1.5Mbps, 3Mbps, 4Mbps
- Support programmable baud rates, even with non-integer clock divider
- Standard asynchronous communication bits (start, stop and parity)
- Support interrupt-based or DMA-based mode
- Support 5-8 bits width transfer

7.2 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

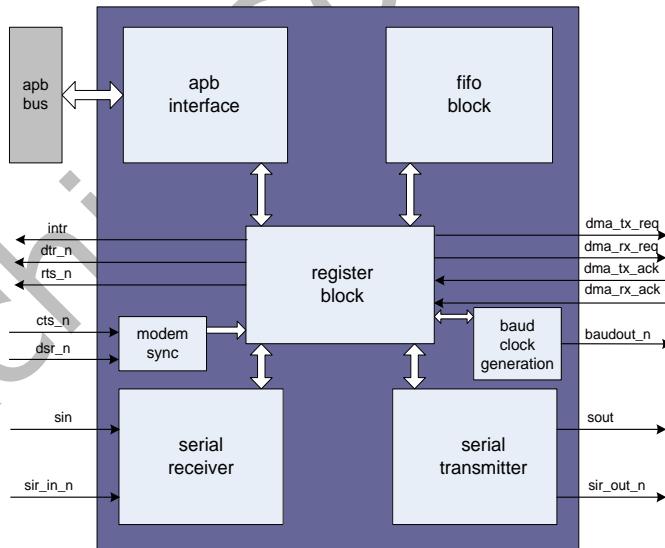


Fig. 5-12 UART Architecture

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

Generates 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.

7.3 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.

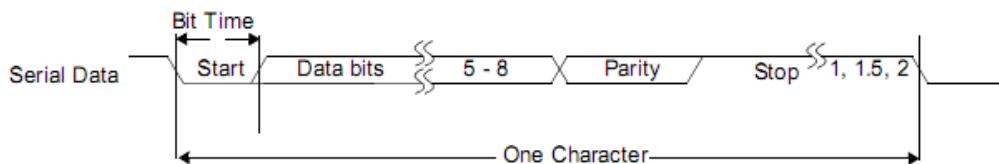


Fig. 5-13 UART Serial protocol

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.

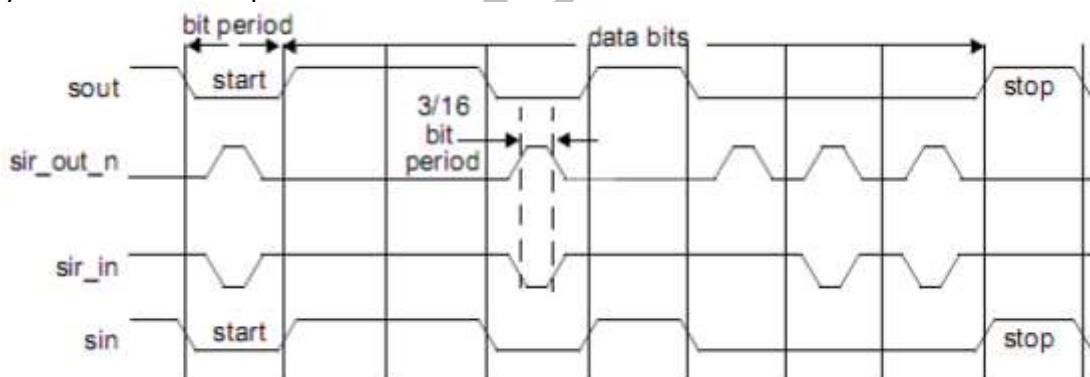


Fig. 5-14 IrDA 1.0

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.

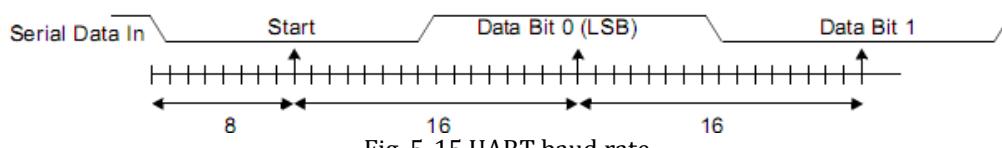


Fig. 5-15 UART baud rate

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 UART0/UART1/UART2 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.

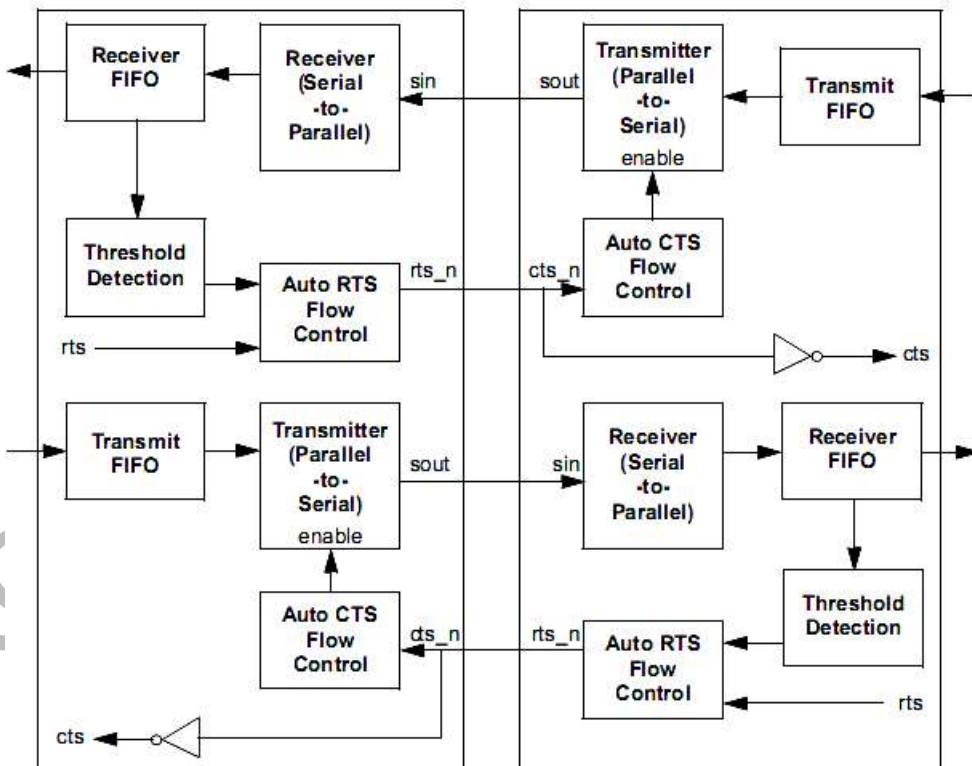


Fig. 5-16 UART Auto flow control block diagram

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)

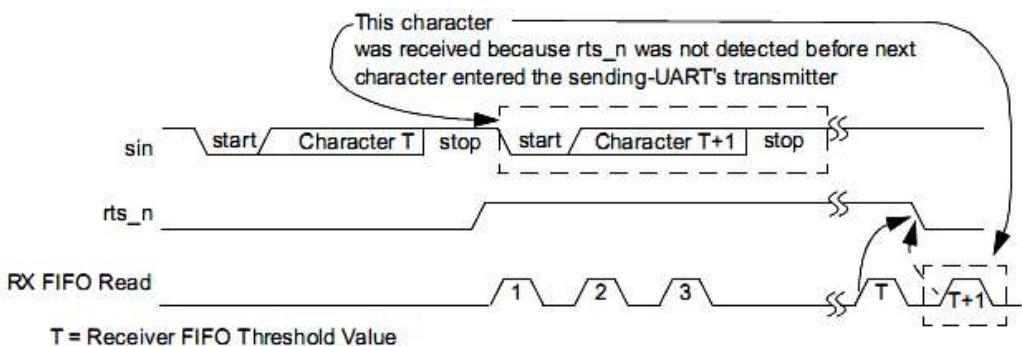


Fig. 5-17 UART AUTO RTS TIMING

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)

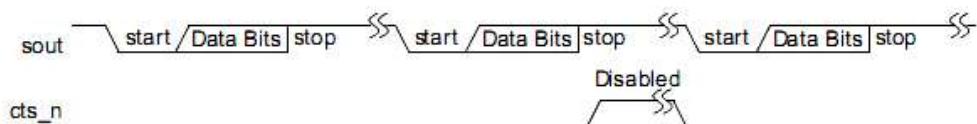


Fig. 5-18 UART AUTO CTS TIMING

7.4 Register Description

This section describes the control/status registers of the design. There are 3 UARTs in RK3228A/RK3228B, and each one has its own base address.

7.4.1 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 | 0x00000000 | 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 | 0x00000000 | Line Status Register |
| UART_MSR | 0x0018 | W | 0x00000000 | Modem Status Register |
| UART_SCR | 0x001c | W | 0x00000000 | Scratchpad Register |
| UART_SRBR | 0x0030 | W | 0x00000000 | Shadow Receive Buffer Register |
| UART_STHR | 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 | 0x00000000 | UART Status Register |
| UART_TFL | 0x0080 | W | 0x00000000 | Transmit FIFO Level |
| UART_RFL | 0x0084 | W | 0x00000000 | Receive FIFO Level |

| Name | Offset | Size | Reset Value | Description |
|------------|--------|------|-------------|-------------------------------|
| 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 |
| 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 | 0x0330372a | UART Component Version |
| UART_CTR | 0x00fc | W | 0x44570110 | Component Type Register |

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

7.4.2 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>data_input 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.</p> <p>If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO is preserved, but any incoming data are lost and an over-run error occurs.</p> |

UART_THR

Address: Operational Base + offset (0x0000)

Transmit Holding Register

| Bit | Attr | Reset Value | Description |
|------|------|-------------|-------------|
| 31:8 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 7:0 | RW | 0x00 | <p>data_output 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)

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:8 | RO | 0x0 | reserved |
| 7:0 | RW | 0x00 | <p>baud_rate_divisor_L 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 freq) / (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)

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:8 | RO | 0x0 | reserved |
| 7 | RW | 0x0 | <p>prog_thre_int_en Programmable THRE Interrupt Mode Enable This is used to enable/disable the generation of THRE Interrupt. 0 = disabled 1 = enabled</p> |
| 6:4 | RO | 0x0 | reserved |
| 3 | RW | 0x0 | <p>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. 0 = disabled 1 = enabled</p> |
| 2 | RW | 0x0 | <p>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. 0 = disabled 1 = enabled</p> |
| 1 | RW | 0x0 | <p>trans_hold_empty_int_en Enable Transmit Holding Register Empty Interrupt.</p> |
| 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. 0 = disabled 1 = enabled</p> |

UART_IIR

Address: Operational Base + offset (0x0008)

Interrupt Identification Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. 00 = disabled 11 = enabled</p> |
| 5:4 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 3:0 | RO | 0x0 | <p>int_id Interrupt ID This indicates the highest priority pending interrupt which can be one of the following types:</p> <ul style="list-style-type: none"> 0000 = modem status 0001 = no interrupt pending 0010 = THR empty 0100 = received data available 0110 = receiver line status 0111 = busy detect 1100 = character timeout |

UART_FCR

Address: Operational Base + offset (0x0008)

FIFO Control Register

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:8 | RO | 0x0 | reserved |
| 7:6 | WO | 0x0 | <p>rcvr_trigger RCVR Trigger. 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"> 00 = 1 character in the FIFO 01 = FIFO 1/4 full 10 = FIFO 1/2 full 11 = FIFO 2 less than ful |
| 5:4 | WO | 0x0 | <p>tx_empty_trigger TX Empty Trigger. 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. The following trigger levels are supported:</p> <ul style="list-style-type: none"> 00 = FIFO empty 01 = 2 characters in the FIFO 10 = FIFO 1/4 full 11 = FIFO 1/2 full |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 3 | WO | 0x0 | dma_mode DMA Mode 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 . 0 = mode 0 1 = mode 11100 = character timeout. |
| 2 | WO | 0x0 | xmit_fifo_reset XMIT FIFO Reset. This resets the control portion of the transmit FIFO and treats the FIFO as empty. This also de-asserts the DMA TX request and single signals when additional DMA handshaking signals are selected . Note that this bit is 'self-clearing'. It is not necessary to clear this bit. |
| 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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 31:8 | RO | 0x0 | reserved |
| 7 | RW | 0x0 | div_lat_access Divisor Latch Access Bit. Writeable 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. |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 6 | RW | 0x0 | <p>break_ctrl Break Control Bit.</p> <p>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.</p> |
| 5 | RO | 0x0 | reserved |
| 4 | RW | 0x0 | <p>even_parity_sel Even Parity Select.</p> <p>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.</p> <p>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.</p> <p>0 = parity disabled 1 = parity enabled</p> |
| 2 | RW | 0x0 | <p>stop_bits_num Number of stop bits.</p> <p>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.</p> <p>0 = 1 stop bit 1 = 1.5 stop bits when DLS (LCR[1:0]) is zero, else 2 stop bit.</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 1:0 | RW | 0x0 | <p>data_length_sel Data Length Select.</p> <p>Writeable 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"> 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits |

UART_MCR

Address: Operational Base + offset (0x0010)

Modem Control Register

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:7 | RO | 0x0 | reserved |
| 6 | RW | 0x0 | <p>sir_mode_en SIR Mode Enable. SIR Mode Enable.</p> <p>This is used to enable/disable the IrDA SIR Mode .</p> <p>0 = IrDA SIR Mode disabled 1 = IrDA SIR Mode enabled</p> |
| 5 | RW | 0x0 | <p>auto_flow_ctrl_en Auto Flow Control Enable.</p> <p>0 = Auto Flow Control Mode disabled 1 = Auto Flow Control Mode enabled</p> |
| 4 | RW | 0x0 | <p>loopback LoopBack Bit.</p> <p>This is used to put the UART into a diagnostic mode for test purposes.</p> |
| 3 | RW | 0x0 | <p>out2 OUT2.</p> <p>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> <p>0 = out2_n de-asserted (logic 1) 1 = out2_n asserted (logic 0)</p> |
| 2 | RW | 0x0 | <p>out1 OUT1</p> <p>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> <p>1'b0: out2_n de-asserted (logic 1) 1'b1: out2_n asserted (logic 0)</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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>0 = dtr_n de-asserted (logic 1) 1 = dtr_n asserted (logic 0)</p> |

UART_LSR

Address: Operational Base + offset (0x0014)

Line Status Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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>0 = no error in RX FIFO 1 = error in RX FIFO</p> |
| 6 | RO | 0x0 | <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> |
| 5 | RO | 0x0 | <p>trans_hold_reg_empty Transmit Holding Register Empty bit.</p> <p>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> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 4 | RO | 0x0 | break_int Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data. |
| 3 | RO | 0x0 | 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. |
| 2 | RO | 0x0 | 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. |
| 1 | RO | 0x0 | 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. |
| 0 | RO | 0x0 | 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. 0 = no data ready 1 = data ready |

UART_MSR

Address: Operational Base + offset (0x0018)

Modem Status Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:8 | RO | 0x0 | reserved |
| 7:0 | RW | 0x00 | temp_store_space This register is for programmers to use as a temporary storage space. |

UART_SRBR

Address: Operational Base + offset (0x0030)

Shadow Receive Buffer Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:8 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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 (0x006c)

Shadow Transmit Holding Register

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 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>0 = FIFO access mode disabled</p> <p>1 = FIFO access mode enabled</p> |

UART_TFR

Address: Operational Base + offset (0x0074)

Transmit FIFO Read

| Bit | Attr | Reset Value | Description |
|------|------|-------------|-------------|
| 31:8 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 7:0 | RO | 0x00 | trans_fifo_read Transmit FIFO Read. 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. |

UART_RFW

Address: Operational Base + offset (0x0078)

Receive FIFO Write

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:10 | RO | 0x0 | reserved |
| 9 | WO | 0x0 | receive_fifo_framing_error Receive FIFO Framing Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). |
| 8 | WO | 0x0 | receive_fifo_parity_error Receive FIFO Parity Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). |
| 7:0 | WO | 0x00 | receive_fifo_write Receive FIFO Write Data. 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. |

UART_USR

Address: Operational Base + offset (0x007c)

UART Status Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:5 | RO | 0x0 | reserved |
| 4 | RO | 0x0 | receive_fifo_full Receive FIFO Full. This is used to indicate that the receive FIFO is completely full. 0 = Receive FIFO not full 1 = Receive FIFO Full This bit is cleared when the RX FIFO is no longer full. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 3 | RO | 0x0 | receive_fifo_not_empty Receive FIFO Not Empty. This is used to indicate that the receive FIFO contains one or more entries. 0 = Receive FIFO is empty 1 = Receive FIFO is not empty This bit is cleared when the RX FIFO is empty. |
| 2 | RO | 0x0 | transn_fifo_empty Transmit FIFO Empty. This is used to indicate that the transmit FIFO is completely empty. 0 = Transmit FIFO is not empty 1 = Transmit FIFO is empty This bit is cleared when the TX FIFO is no longer empty |
| 1 | RO | 0x0 | trans_fifo_not_full Transmit FIFO Not Full. This is used to indicate that the transmit FIFO is not full. 0 = Transmit FIFO is full 1 = Transmit FIFO is not full This bit is cleared when the TX FIFO is full. |
| 0 | RO | 0x0 | 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. 0 = UART is idle or inactive 1 = UART is busy (actively transferring data) |

UART_TFL

Address: Operational Base + offset (0x0080)

Transmit FIFO Level

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:5 | RO | 0x0 | reserved |
| 4:0 | RW | 0x00 | trans_fifo_level Transmit FIFO Level. This indicates the number of data entries in the transmit FIFO. |

UART_RFL

Address: Operational Base + offset (0x0084)

Receive FIFO Level

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:5 | RO | 0x0 | reserved |
| 4:0 | RO | 0x00 | receive_fifo_level Receive FIFO Level. This indicates the number of data entries in the receive FIFO. |

UART_SRR

Address: Operational Base + offset (0x0088)

Software Reset Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:3 | RO | 0x0 | reserved |
| 2 | WO | 0x0 | xmit_fifo_reset XMIT FIFO Reset. This is a shadow register for the XMIT FIFO Reset bit (FCR[2]). |
| 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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:1 | RO | 0x0 | reserved |

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:1 | RO | 0x0 | reserved |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:1 | RO | 0x0 | reserved |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. 0 = Halt TX disabled 1 = Halt TX enabled |

UART_DMASA

Address: Operational Base + offset (0x00a8)

DMA Software Acknowledge

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:1 | RO | 0x0 | reserved |
| 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_CPR

Address: Operational Base + offset (0x00f4)

Component Parameter Register

UART_CPR is UART0's own unique register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:24 | RO | 0x0 | reserved |
| 23:16 | RO | 0x00 | FIFO_MODE 0x00 = 0 0x01 = 16 0x02 = 32 to 0x80 = 2048 0x81- 0xff = reserved |
| 15:14 | RO | 0x0 | reserved |
| 13 | RO | 0x0 | DMA_EXTRA 0 = FALSE 1 = TRUE |
| 12 | RO | 0x0 | UART_ADD_ENCODED_PARAMS 0 = FALSE 1 = TRUE |
| 11 | RO | 0x0 | SHADOW 0 = FALSE 1 = TRUE |
| 10 | RO | 0x0 | FIFO_STAT 0 = FALSE 1 = TRUE |
| 9 | RO | 0x0 | FIFO_ACCESS 0 = FALSE 1 = TRUE |
| 8 | RO | 0x0 | NEW_FEAT 0 = FALSE 1 = TRUE |
| 7 | RO | 0x0 | SIR_LP_MODE 0 = FALSE 1 = TRUE |
| 6 | RO | 0x0 | SIR_MODE 0 = FALSE 1 = TRUE |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 5 | RO | 0x0 | THRE_MODE 0 = FALSE 1 = TRUE |
| 4 | RO | 0x0 | AFCE_MODE 0 = FALSE 1 = TRUE |
| 3:2 | RO | 0x0 | reserved |
| 1:0 | RO | 0x0 | APB_DATA_WIDTH 00 = 8 bits 01 = 16 bits 10 = 32 bits 11 = reserved |

UART_UCV

Address: Operational Base + offset (0x00f8)

UART Component Version

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:0 | RO | 0x0330372a | 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. |

7.5 Interface Description

Table 5-9 UART Interface Description

| Module pin | Dir | Pad name | IOMUX |
|-------------------------|-----|---------------------------------|-------------------------------|
| UART0 Interface | | | |
| uart0_sin | I | IO_UART0sin_GPIO2d3 | GRF_GPIO2D_IOMUX[7:6]=2'b01 |
| uart0_sout | O | IO_UART0sout_GPIO2d2 | GRF_GPIO2D_IOMUX[5:4]=2'b01 |
| uart0_cts_n | I | IO_UART0ctsn_GPIO2d5 | GRF_GPIO2D_IOMUX[11:10]=2'b01 |
| uart0_rts_n | O | IO_UART0rtsn_CLKout1_GPIO0c1 | GRF_GPIO0C_IOMUX[3:2]=2'b01 |
| UART1 Interface | | | |
| uart1_sin | I | IO_UART1sin_UART21sin_GPIO1b2 | GRF_GPIO1B_IOMUX[5:4]=2'b01 |
| uart1_sout | O | IO_UART1sout_UART21sout_GPIO1b1 | GRF_GPIO1B_IOMUX[3:2]=2'b01 |
| uart1_cts_n | I | IO_UART1ctsn_CLKOUT32k_GPIO1b0 | GRF_GPIO1B_IOMUX[1:0]=2'b01 |
| uart1_rts_n | O | IO_UART1rtsn_PWM1ir_GPIO1b3 | GRF_GPIO1B_IOMUX[7:6]=2'b01 |
| UART11 Interface | | | |
| uart11_sin | I | IO_UART11sin_GPIO3b5 | GRF_GPIO3B_IOMUX[11:10]=2'b01 |
| uart11_sout | O | IO_UART11sout_GPIO3b6 | GRF_GPIO3B_IOMUX[13:12]=2'b01 |

| | | | |
|-------------------------|---|---------------------------------|-------------------------------|
| uart11_cts_n | I | IO_UART11ctsn_GPIO3a7 | GRF_GPIO3A_IOMUX[15:14]=2'b01 |
| uart11_rts_n | O | IO_UART11rtsn_GPIO3a6 | GRF_GPIO3A_IOMUX[13:12]=2'b01 |
| UART2 Interface | | | |
| uart2_sin | I | IO_SDMMCd1_UART2sin_GPIO1c3 | GRF_GPIO1C_IOMUX[7:6]=2'b10 |
| uart2_sout | O | IO_SDMMCd0_UART2sout_GPIO1c2 | GRF_GPIO1C_IOMUX[5:4]=2'b10 |
| uart2_cts_n | I | IO_UART2ctsn_GPIO0d1 | GRF_GPIO0D_IOMUX[3:2]=2'b01 |
| uart2_rts_n | O | IO_UART2rtsn_TSADCshut_GPIO0d0 | GRF_GPIO0D_IOMUX[1:0]=2'b01 |
| UART21 Interface | | | |
| uart21_sin | I | IO_UART1sin_UART21sin_GPIO1b2 | GRF_GPIO1B_IOMUX[5:4]=2'b10 |
| uart21_sout | O | IO_UART1sout_UART21sout_GPIO1b1 | GRF_GPIO1B_IOMUX[3:2]=2'b10 |

The I/O interface of UART1 can be chosen by setting GRF_CON_IOMUX[11](uart1sel) bit, if this bit is set to 1, UART1 uses the UART11 I/O interface. The I/O interface of UART2 can be chosen by setting GRF_CON_IOMUX[8](uart2sel) bit, if this bit is set to 1, UART2 uses the UART21 I/O interface.

7.6 Application Notes

7.6.1 None FIFO Mode Transfer Flow

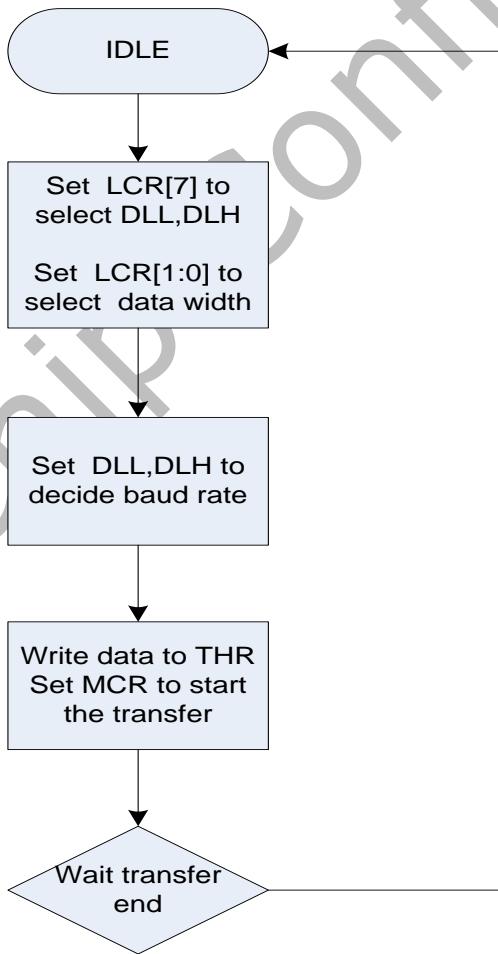


Fig. 5-19 UART none fifo mode

7.6.2 FIFO Mode Transfer Flow

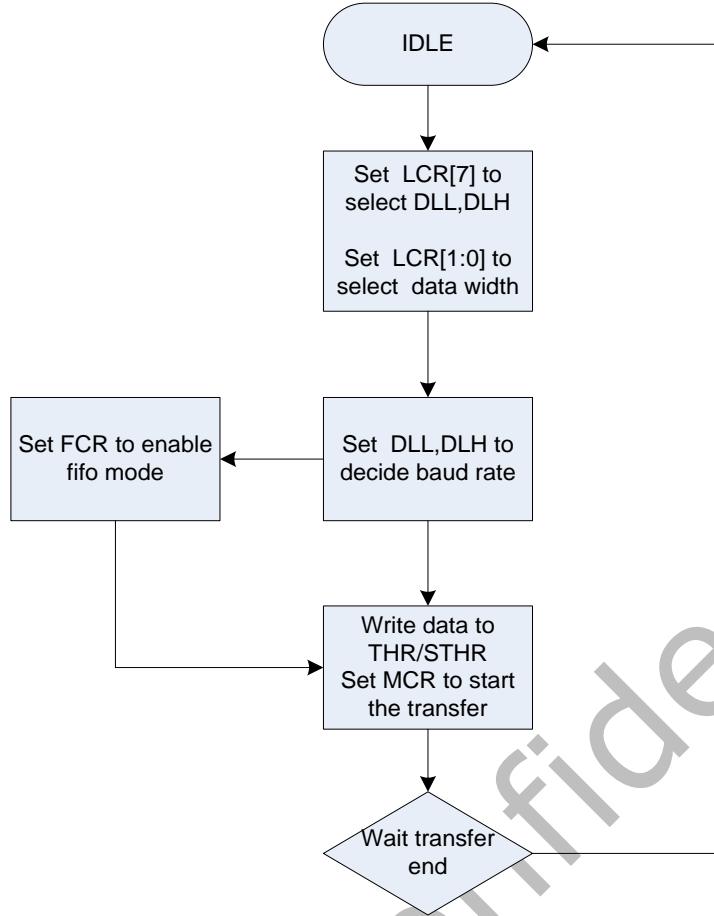


Fig. 5-20 UART fifo mode

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.

7.6.3 Baud Rate Calculation

UART clock generation

The following figures shows the UART clock generation.

UART0, UART1 and UART2 source clocks can be selected from three PLL outputs (CODEC PLL/GENERAL PLL/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.

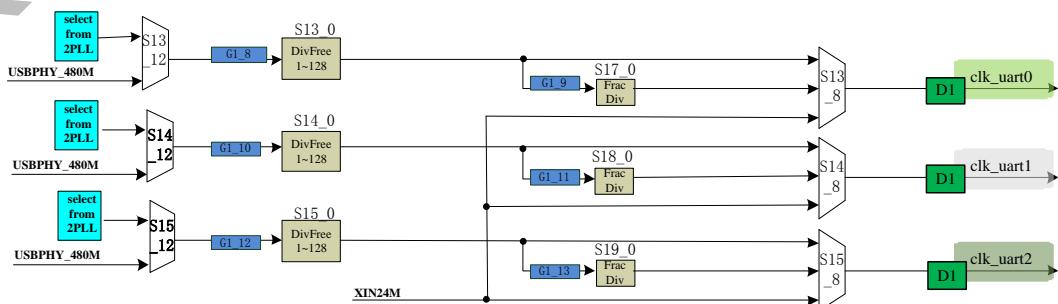


Fig. 5-21 UART clock generation

UART baud rate configuration

The following table provides some reference configuration for different UART baud rates.

Table 5-10 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; Configure 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 |

1.6.4 CTS_n and RTS_n Polarity Configurable

The polarity of cts_n and rts_n ports can be configured by GRF registers.

- GRF_SOC_CON2[2:0] (grf_uart_cts_sel[2:0]) used to configure the polarity of cts_n.Every bit for one UART, bit2 is for UART2,bit1 is for UART1, bit0 is for UART0.
- GRF_SOC_CON2[7:5] (grf_uart_rts_sel[4:0]) used to configure the polarity of rts_n.Every bit for one UART, bit2 is for UART2,bit1 is for UART1, bit0 is for UART0.
- 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 8 GPIO

8.1 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

8.2 Block Diagram

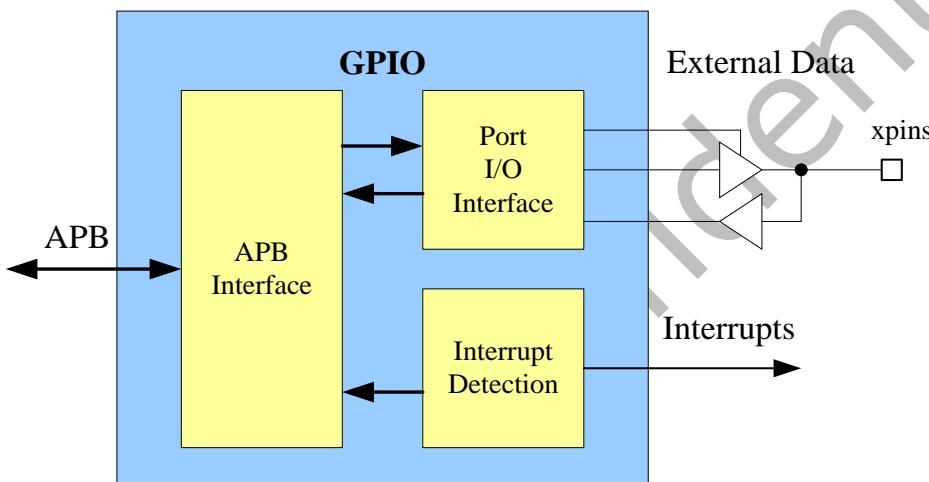


Fig. 5-22GPIO block diagram

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.

8.3 Function Description

8.3.1 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.

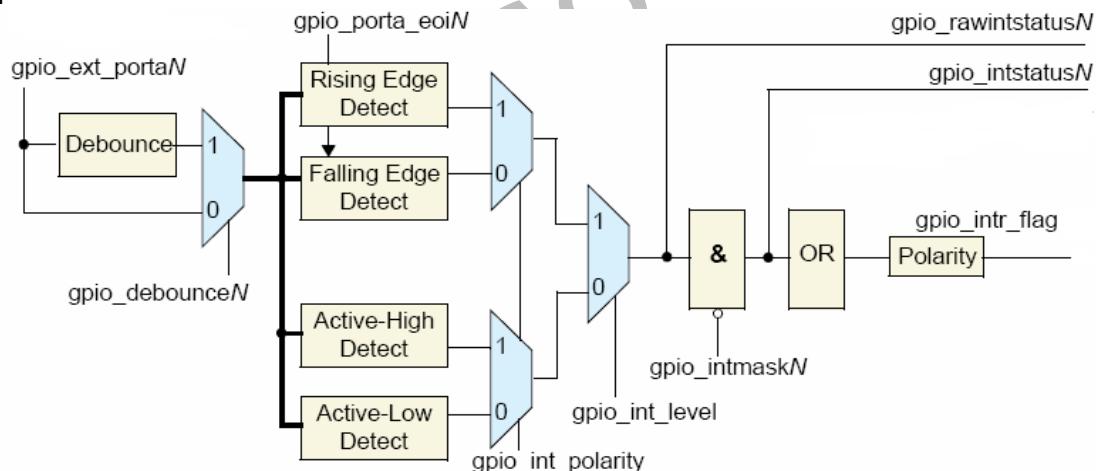


Fig. 5-23 GPIO Interrupt RTL Block Diagram

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).

8.3.2 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.

GPIOs' hierarchy in the chip

GPIO0, GPIO1, GPIO2, GPIO3 are in PD_BUS subsystem.

8.4 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 4 GPIOs (GPIO0 ~ GPIO3), and each of them has same register group. Therefore, 4 GPIOs' register groups have 4 different base addresses.

8.4.1 Registers Summary

| Name | Offset | Size | Reset Value | Description |
|--------------------|--------|------|-------------|---|
| GPIO_SWPORTA_DR | 0x0000 | W | 0x00000000 | Port A data register |
| GPIO_SWPORTA_DDR | 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_INTTYPE_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

8.4.2 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 |
|------------|-------------|--------------------|---|
| 31:0 | RW | 0x00000000 | gpio_swporta_ddr Values written to this register independently control the direction of the corresponding data bit in Port A. 0: Input (default) 1: Output |

GPIO_INTEN

Address: Operational Base + offset (0x0030)

Interrupt enable register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RW | 0x00000000 | gpio_int_en 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. Interrupts are disabled on the corresponding bits of Port A if the corresponding data direction register is set to Output. 0: Configure Port A bit as normal GPIO signal (default) 1: Configure Port A bit as interrupt |

GPIO_INTMASK

Address: Operational Base + offset (0x0034)

Interrupt mask register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RW | 0x00000000 | gpio_int_mask 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. 0: Interrupt bits are unmasked (default) 1: Mask interrupt |

GPIO_INTTYPE_LEVEL

Address: Operational Base + offset (0x0038)

Interrupt level register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RW | 0x00000000 | gpio_inctype_level Controls the type of interrupt that can occur on Port A. 0: Level-sensitive (default) 1: Edge-sensitive |

GPIO_INT_POLARITY

Address: Operational Base + offset (0x003c)

Interrupt polarity register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RW | 0x00000000 | gpio_int_polarity Controls the polarity of edge or level sensitivity that can occur on input of Port A. 0: Active-low (default) 1: Active-high |

GPIO_INT_STATUS

Address: Operational Base + offset (0x0040)

Interrupt status of port A

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. 0: No debounce (default) 1: Enable debounce |

GPIO_PORTA_EOI

Address: Operational Base + offset (0x004c)

Port A clear interrupt register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. 0: No interrupt clear (default) 1: Clear interrupt |

GPIO_EXT_PORTA

Address: Operational Base + offset (0x0050)

Port A external port register

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. 0: No synchronization to pclk_intr (default) 1: Synchronize to pclk_intr |

8.5 Interface Description

Table 5-11 GPIO interface description

| Module Pin | Dir | Pad Name | IOMUX Setting |
|------------------------|------------|-----------------|------------------------------|
| GPIO0 Interface | | | |
| gpio0_porta[3:0] | I/O | GPIO0_A[3:0] | GRF_GPIO0A_IOMUX[7:0]=8'h0 |
| gpio0_porta[7:6] | | GPIO0_A[7:6] | GRF_GPIO0A_IOMUX[15:12]=4'h0 |
| gpio0_porta[9:8] | I/O | GPIO0_B[1:0] | GRF_GPIO0B_IOMUX[3:0]=4'h0 |
| gpio0_porta[14:11] | | GPIO0_B[7:3] | GRF_GPIO0B_IOMUX[15:6]=10'h0 |
| gpio0_porta[20:16] | I/O | GPIO0_C[4:0] | GRF_GPIO0C_IOMUX[9:0]=10'h0 |
| gpio0_porta[23:22] | | GPIO0_C[7:6] | GRF_GPIO0C_IOMUX[15:12]=4'h0 |
| gpio0_porta[31:24] | I/O | GPIO0_D[7:0] | GRF_GPIO0D_IOMUX[15:0]=16'h0 |
| GPIO1 Interface | | | |
| gpio1_porta[5:0] | I/O | GPIO1_A[5:0] | GRF_GPIO1A_IOMUX[11:0]=12'h0 |
| gpio1_porta[7] | | GPIO1_A[7] | GRF_GPIO1A_IOMUX[15:14]=2'h0 |
| gpio1_porta[12:8] | I/O | GPIO1_B[4:0] | GRF_GPIO1B_IOMUX[9:0]=10'h0 |
| gpio1_porta[15:14] | | GPIO1_B[7:6] | GRF_GPIO1B_IOMUX[15:12]=4'h0 |
| gpio1_porta[23:16] | I/O | GPIO1_C[7:0] | GRF_GPIO1C_IOMUX[15:0]=16'h0 |
| gpio1_porta[31:24] | I/O | GPIO1_D[7:0] | GRF_GPIO1D_IOMUX[15:0]=16'h0 |
| GPIO2 Interface | | | |
| gpio2_porta[7:0] | I/O | GPIO2_A[7:0] | GRF_GPIO2A_IOMUX[15:0]=16'h0 |
| gpio2_porta[15:8] | I/O | GPIO2_B[7:0] | GRF_GPIO2B_IOMUX[15:0]=16'h0 |
| gpio2_porta[23:16] | I/O | GPIO2_C[7:0] | GRF_GPIO2C_IOMUX[15:0]=16'h0 |
| gpio2_porta[29:24] | I/O | GPIO2_D[5:0] | GRF_GPIO2D_IOMUX[11:0]=12'h0 |

| Module Pin | Dir | Pad Name | IOMUX Setting |
|------------------------|-----|--------------|------------------------------|
| GPIO3 Interface | | | |
| gpio3_porta[7:0] | I/O | GPIO3_A[7:0] | GRF_GPIO3A_IOMUX[15:0]=16'h0 |
| gpio3_porta[8] | I/O | GPIO3_B[0] | GRF_GPIO3B_IOMUX[1:0] =2'h0 |
| gpio3_porta[15:11] | | GPIO3_B[7:3] | GRF_GPIO3B_IOMUX[15:6]=10'h0 |
| gpio3_porta[23:16] | I/O | GPIO3_C[7:0] | GRF_GPIO3C_IOMUX[15:0]=16'h0 |
| gpio3_porta[31:25] | I/O | GPIO3_D[7:1] | GRF_GPIO3D_IOMUX[15:2]=14'h0 |

8.6 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_INTTYPE_LEVEL[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!

Chapter 9 I2C Interface

9.1 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

9.2 Block Diagram

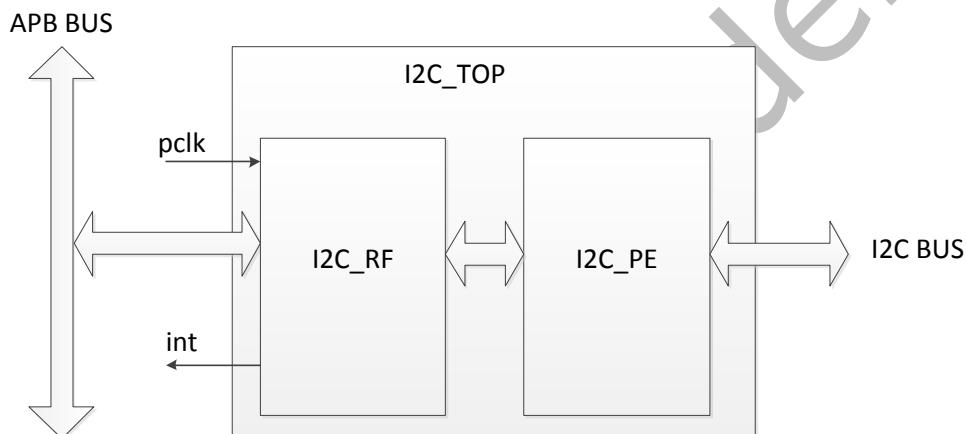


Fig. 5-24 I2C architecture

9.2.1 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.

9.2.2 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.

9.2.3 I2C_TOP

I2C_TOP module is the top module of the I2C controller.

9.3 Function Description

This chapter provides a description about the functions and behavior under various conditions. The I2C controller supports only Masterfunction. 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.

9.3.1 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 and 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 working clock so the APB clock will determine the I2C bus clock. The correct register setting is subject to the system requirement.

9.3.2 Master Mode Programming

- 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:

$$\text{SCL Divisor} = 8 * (\text{CLKDIVL} + 1 + \text{CLKDIVH} + 1)$$

SCL = PCLK/ SCLK Divisor

- Data Receiver Register Access

When the I2C controller received MRXCNT bytes data, CPU can get the data through register RXDATA0 ~ RXDATA7. The controller can receive up to 32 bytes' data in one transaction.

When MRXCNT register is written, the I2C controller will start to drive SCL to receive data.

- Transmit Transmitter Register

Data to transmit are written to TXDATA0~7 by CPU. The controller can transmit up to 32 bytes' data in one transaction. The lower byte will be transmitted first.

When MTXCNT register is written, the I2C controller will start to transmit data.

- Start Command

Write 1 to I2C_CON[3], the controller will send I2C start command.

- Stop Command

Write 1 to I2C_CON[4], the controller will send I2C stop command

- 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 trigger 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.

- 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].

- 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 completed transmitting a byte.
- Byte received finish interrupt (Bit 1): The bit is asserted when Master completed receiving a byte.
- MTXCNT bytes data transmitted finish interrupt (Bit 2): The bit is asserted when

- Master completed transmitting MTXCNT bytes.
- MRXCNT bytes data received finish interrupt (Bit 3): The bit is asserted when Master completed receiving MRXCNT bytes.
- Start interrupt (Bit 4): The bit is asserted when Master finished asserting start command to I2C bus.
- Stop interrupt (Bit 5): The bit is asserted when Master finished asserting stop command to I2C bus.
- NAK received interrupt (Bit 6): The bit is asserted when Master received a NAK handshake.
- 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.
- 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.
- I2C controller data transfer waveform
 - Bit transferring
 - ◆ 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.

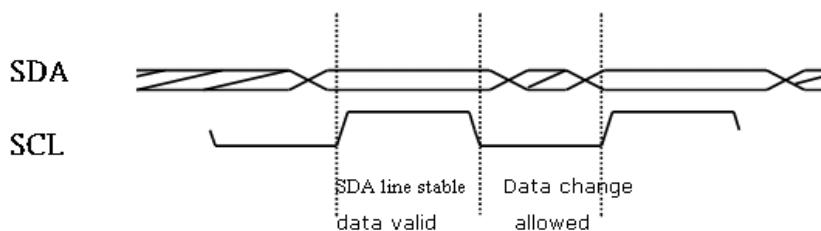


Fig. 5-25 I2C DATA Validity

- ◆ 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.

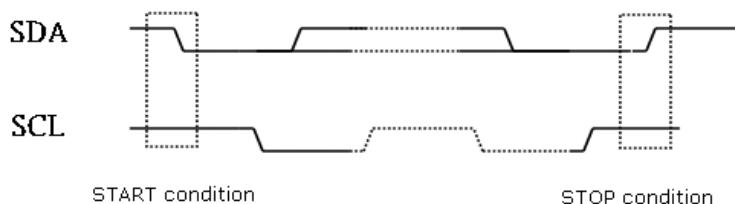


Fig. 5-26 I2C Start and stop conditions

- ◆ Data transfer
 - Acknowledge

After a byte of data transferring (clocks labeled as 1~8), in 9th 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".

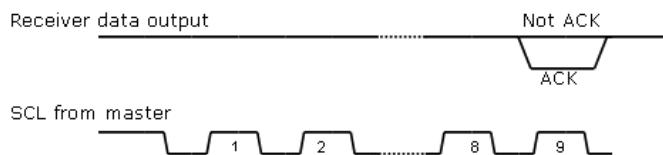


Fig. 5-27 I2C Acknowledge

➤ 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.

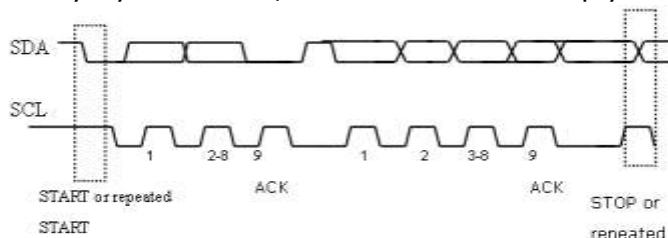


Fig. 5-28 I2C byte transfer

9.4 Register Description

9.4.1 Registers Summary

| Name | Offset | Size | Reset Value | Description |
|----------------|--------|------|-------------|--|
| RKI2C_CON | 0x0000 | W | 0x00000000 | control register |
| RKI2C_CLKDIV | 0x0004 | W | 0x00000001 | clock divider register |
| RKI2C_MRXADDR | 0x0008 | W | 0x00000000 | the slave address accessed for master rx mode |
| RKI2C_MRXRADDR | 0x000c | W | 0x00000000 | the slave register address accessed for master rx mode |
| RKI2C_MTXCNT | 0x0010 | W | 0x00000000 | master transmit count |
| RKI2C_MRXCNT | 0x0014 | W | 0x00000000 | master rx count |
| RKI2C_IEN | 0x0018 | W | 0x00000000 | interrupt enable register |
| RKI2C_IPD | 0x001c | W | 0x00000000 | interrupt pending register |
| RKI2C_FCNT | 0x0020 | W | 0x00000000 | finished count |
| RKI2C_TXDATA0 | 0x0100 | W | 0x00000000 | I2C tx data register 0 |
| RKI2C_TXDATA1 | 0x0104 | W | 0x00000000 | I2C tx data register 1 |
| RKI2C_TXDATA2 | 0x0108 | W | 0x00000000 | I2C tx data register 2 |
| RKI2C_TXDATA3 | 0x010c | W | 0x00000000 | I2C tx data register 3 |
| RKI2C_TXDATA4 | 0x0110 | W | 0x00000000 | I2C tx data register 4 |
| RKI2C_TXDATA5 | 0x0114 | W | 0x00000000 | I2C tx data register 5 |
| RKI2C_TXDATA6 | 0x0118 | W | 0x00000000 | I2C tx data register 6 |
| RKI2C_TXDATA7 | 0x011c | W | 0x00000000 | I2C tx data register 7 |
| RKI2C_RXDATA0 | 0x0200 | W | 0x00000000 | I2C rx data register 0 |
| RKI2C_RXDATA1 | 0x0204 | W | 0x00000000 | I2C rx data register 1 |
| RKI2C_RXDATA2 | 0x0208 | W | 0x00000000 | I2C rx data register 2 |
| RKI2C_RXDATA3 | 0x020c | W | 0x00000000 | I2C rx data register 3 |
| RKI2C_RXDATA4 | 0x0210 | W | 0x00000000 | I2C rx data register 4 |
| RKI2C_RXDATA5 | 0x0214 | W | 0x00000000 | I2C rx data register 5 |

| Name | Offset | Size | Reset Value | Description |
|---------------|--------|------|-------------|------------------------|
| RKI2C_RXDATA6 | 0x0218 | W | 0x00000000 | I2C rx data register 6 |
| RKI2C_RXDATA7 | 0x021c | W | 0x00000000 | I2C rx data register 7 |

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

9.4.2 Detail Register Description

RKI2C_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 |
| 5 | RW | 0x0 | ack last byte acknowledge control in master receive mode 1'b0: ACK 1'b1: NAK |
| 4 | RW | 0x0 | stop stop enable stop enable, when this bit is written to 1, I2C will generate stop signal. |
| 3 | RW | 0x0 | start start enable start enable, when this bit is written to 1, I2C will generate start signal. |
| 2:1 | RW | 0x0 | i2c_mode i2c mode select 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 1'b0: not enable 1'b1: enable |

RKI2C_CLKDIV

Address: Operational Base + offset (0x0004)
clock divider register

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:16 | RW | 0x0000 | CLKDIVH scl high level clock count $T(SCL_HIGH) = T(PCLK) * (CLKDIVH + 1) * 8$ |
| 15:0 | RW | 0x0001 | CLKDIVL scl low level clock count $T(SCL_LOW) = T(PCLK) * (CLKDIVL + 1) * 8$ |

RKI2C_MRXADDR

Address: Operational Base + offset (0x0008)
the slave address accessed for master rx mode

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:27 | RO | 0x0 | reserved |
| 26 | RW | 0x0 | addhvld address high byte valid 1'b0:invalid 1'b1:valid |
| 25 | RW | 0x0 | addmvld address middle byte valid 1'b0:invalid 1'b1:valid |
| 24 | RW | 0x0 | addlvld address low byte valid 1'b0:invalid 1'b1:valid |
| 23:0 | RW | 0x0000000 | saddr master address register the lowest bit indicate write or read 24 bits address register |

RKI2C_MRXRADDR

Address: Operational Base + offset (0x000c)
the slave register address accessed for master rx mode

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:27 | RO | 0x0 | reserved |
| 26 | RW | 0x0 | sraddhvld address high byte valid 1'b0:invalid 1'b1:valid |
| 25 | RW | 0x0 | sraddmvld address middle byte valid 1'b0:invalid 1'b1:valid |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 24 | RW | 0x0 | sraaddrvld address low byte valid 1'b0:invalid 1'b1:valid |
| 23:0 | RW | 0x000000 | sraaddr slave register address accessed 24 bits register address |

RKI2C_MTXCNT

Address: Operational Base + offset (0x0010)

master transmit count

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:6 | RO | 0x0 | reserved |
| 5:0 | RW | 0x00 | mtxcnt master transmit count 6 bits counter |

RKI2C_MRXCNT

Address: Operational Base + offset (0x0014)

masterrx count

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:6 | RO | 0x0 | reserved |
| 5:0 | RW | 0x00 | mrxcnt master rx count 6 bits counter |

RKI2C_IEN

Address: Operational Base + offset (0x0018)

interrupt enable register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:7 | RO | 0x0 | reserved |
| 6 | RW | 0x0 | nakrcvien NAK handshake received interrupt enable 1'b0:disable 1'b1:enable |
| 5 | RW | 0x0 | stopien stop operation finished interrupt enable 1'b0:disable 1'b1:enable |
| 4 | RW | 0x0 | startien start operation finished interrupt enable 1'b0:disable 1'b1:enable |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 3 | RW | 0x0 | mbrfien MRXCNT data received finished interrupt enable 1'b0:disable 1'b1:enable |
| 2 | RW | 0x0 | mbtfien MTXCNT data transfer finished interrupt enable 1'b0:disable 1'b1:enable |
| 1 | RW | 0x0 | brfien byte rx finished interrupt enable 1'b0:disable 1'b1:enable |
| 0 | RW | 0x0 | btfien byte tx finished interrupt enable 1'b0:disable 1'b1:enable |

RKI2C_IPD

Address: Operational Base + offset (0x001c)
 interrupt pending register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:7 | RO | 0x0 | reserved |
| 6 | W1 C | 0x0 | nakrcvipd NAK handshake received interrupt pending bit 1'b0:no interrupt available 1'b1:NAK handshake received interrupt appear, write 1 to clear |
| 5 | W1 C | 0x0 | stopipd stop operation finished interrupt pending bit 1'b0:no interrupt available 1'b1:stop operation finished interrupt appear, write 1 to clear |
| 4 | W1 C | 0x0 | startipd start operation finished interrupt pending bit 1'b0:no interrupt available 1'b1:start operation finished interrupt appear, write 1 to clear |
| 3 | W1 C | 0x0 | mbrfidp MRXCNT data received finished interrupt pending bit 1'b0:no interrupt available 1'b1:MRXCNT data received finished interrupt appear, write 1 to clear |
| 2 | W1 C | 0x0 | mbtfidp MTXCNT data transfer finished interrupt pending bit 1'b0:no interrupt available 1'b1:MTXCNT data transfer finished interrupt appear, write 1 to clear |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 1 | W1 C | 0x0 | brfidp byte rx finished interrupt pending bit 1'b0: no interrupt available 1'b1: byte rx finished interrupt appear, write 1 to clear |
| 0 | W1 C | 0x0 | btfidp byte tx finished interrupt pending bit 1'b0: no interrupt available 1'b1: byte tx finished interrupt appear, write 1 to clear |

RKI2C_FCNT

Address: Operational Base + offset (0x0020)

finished count

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:6 | RO | 0x0 | reserved |
| 5:0 | RO | 0x00 | fcnt finished count the count of data which has been transmitted or received for debug purpose |

RKI2C_TXDATA0

Address: Operational Base + offset (0x0100)

I2C tx data register 0

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RW | 0x00000000 | txdata0 data0 to be transmitted 32 bits data |

RKI2C_TXDATA1

Address: Operational Base + offset (0x0104)

I2C tx data register 1

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RW | 0x00000000 | txdata1 data1 to be transmitted 32 bits data |

RKI2C_TXDATA2

Address: Operational Base + offset (0x0108)

I2C tx data register 2

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RW | 0x00000000 | txdata2 data2 to be transmitted 32 bits data |

RKI2C_TXDATA3

RK3228A/RK3228B TRM

Address: Operational Base + offset (0x010c)

I2C tx data register 3

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RW | 0x00000000 | txdata3 data3 to be transmitted 32 bits data |

RKI2C_TXDATA4

Address: Operational Base + offset (0x0110)

I2C tx data register 4

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RW | 0x00000000 | txdata4 data4 to be transmitted 32 bits data |

RKI2C_TXDATA5

Address: Operational Base + offset (0x0114)

I2C tx data register 5

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RW | 0x00000000 | txdata5 data5 to be transmitted 32 bits data |

RKI2C_TXDATA6

Address: Operational Base + offset (0x0118)

I2C tx data register 6

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RW | 0x00000000 | txdata6 data6 to be transmitted 32 bits data |

RKI2C_TXDATA7

Address: Operational Base + offset (0x011c)

I2C tx data register 7

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RW | 0x00000000 | txdata7 data7 to be transmitted 32 bits data |

RKI2C_RXDATA0

Address: Operational Base + offset (0x0200)

I2C rx data register 0

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RO | 0x00000000 | rxdata0 data0 received 32 bits data |

RKI2C_RXDATA1

Address: Operational Base + offset (0x0204)

I2C rx data register 1

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RO | 0x00000000 | rxdata1 data1 received 32 bits data |

RKI2C_RXDATA2

Address: Operational Base + offset (0x0208)

I2C rx data register 2

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RO | 0x00000000 | rxdata2 data2 received 32 bits data |

RKI2C_RXDATA3

Address: Operational Base + offset (0x020c)

I2C rx data register 3

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RO | 0x00000000 | rxdata3 data3 received 32 bits data |

RKI2C_RXDATA4

Address: Operational Base + offset (0x0210)

I2C rx data register 4

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RO | 0x00000000 | rxdata4 data4 received 32 bits data |

RKI2C_RXDATA5

Address: Operational Base + offset (0x0214)

I2C rx data register 5

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RO | 0x00000000 | rxdata5 data5 received 32 bits data |

RKI2C_RXDATA6

Address: Operational Base + offset (0x0218)

I2C rx data register 6

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RO | 0x00000000 | rxdata6 data6 received 32 bits data |

RKI2C_RXDATA7

Address: Operational Base + offset (0x021c)

I2C rx data register 7

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RO | 0x00000000 | rxdata7 data7 received 32 bits data |

9.5 Interface Description

Table 5-12 I2C Interface Description

| Module pin | Direction | Pad name | IOMUX |
|-----------------------|------------------|----------------------------|-------------------------------|
| I2C0 Interface | | | |
| i2c0_sda | I/O | IO_I2C0pmusda_PWM2_GPIO0a1 | GRF_GPIO0A_IOMUX[3:2]=2'b01 |
| i2c0_scl | I/O | IO_I2C0pmuscl_PWM1_GPIO0a0 | GRF_GPIO0A_IOMUX[1:0]=2'b01 |
| I2C1 Interface | | | |
| i2c1_sda | I/O | IO_I2C1tpsda_GPIO0a3 | GRF_GPIO0A_IOMUX[7:6]=2'b01 |
| i2c1_scl | I/O | IO_I2C1tpsc1_GPIO0a2 | GRF_GPIO0A_IOMUX[5:4]=2'b01 |
| I2C2 Interface | | | |
| i2c2_sda | I/O | IO_I2C2sda_GPIO2c4 | GRF_GPIO0A_IOMUX[9:8]=2'b01 |
| i2c2_scl | I/O | IO_I2C2scl_GPIO2c5 | GRF_GPIO0A_IOMUX[11:10]=2'b01 |

9.6 Application 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, and mix mode. Users are strongly advised to follow

- Transmit only mode (I2C_CON[1:0]=2'b00)

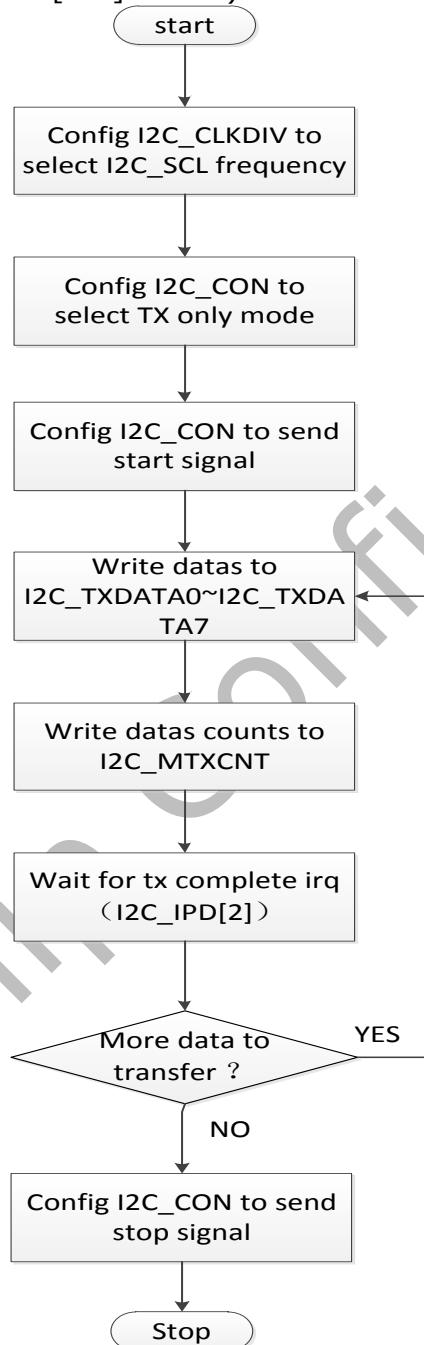


Fig. 5-29 I2C Flow chat for transmit only mode

- Receive only mode (I2C_CON[1:0]=2'b10)

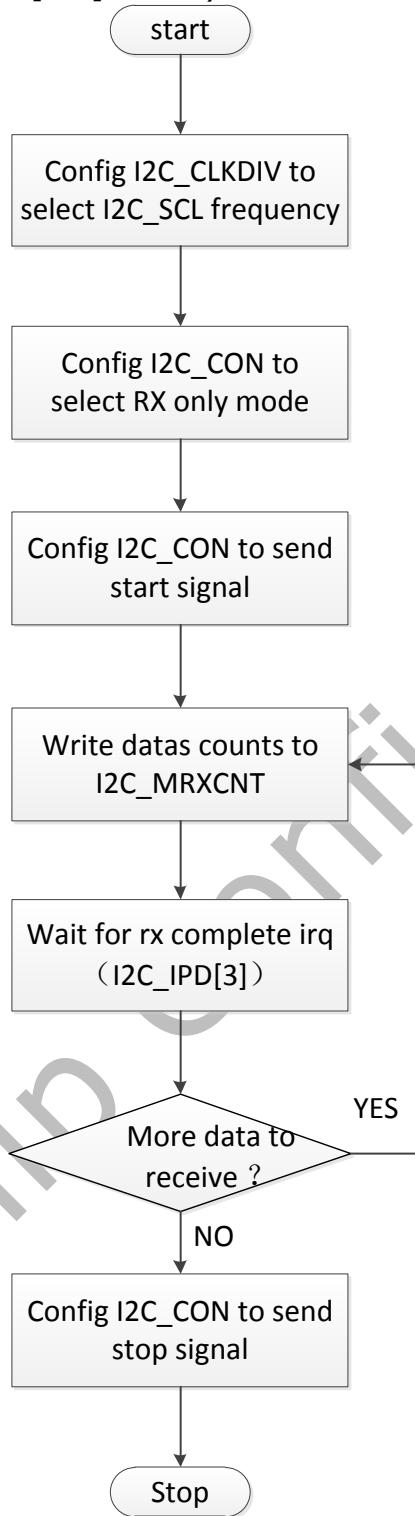


Fig. 5-30 I2C Flow chat for receive only mode

- Mix mode ($\text{I2C_CON}[1:0]=2'b01$ or $\text{I2C_CON}[1:0]=2'b11$)

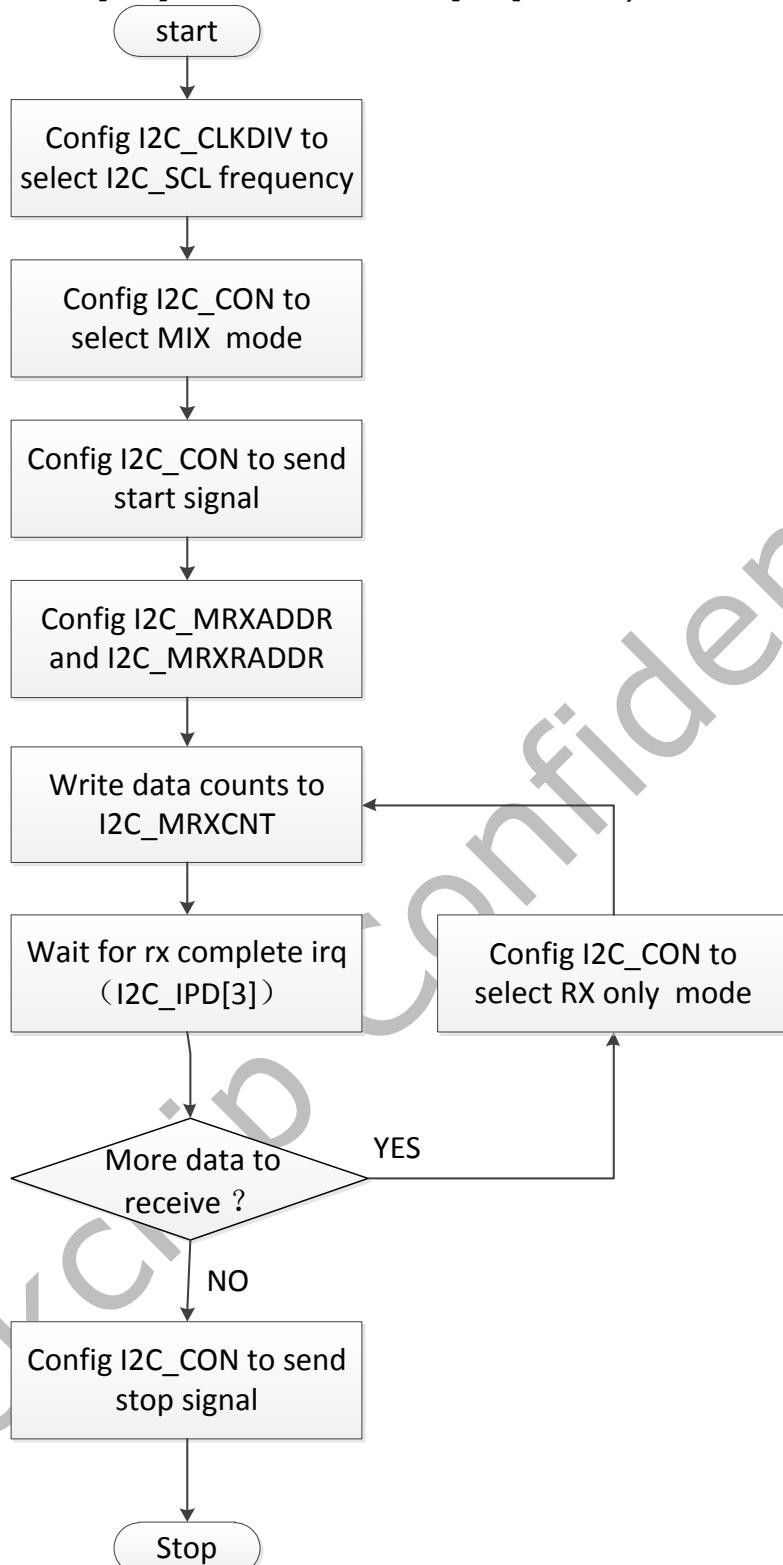


Fig. 5-31 I2C Flow chat for mix mode

Chapter 10 I2S/PCM Controller

10.1 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 and stereo input are supported in I2S/PCM controller.

There are three I2S/PCM controllers embedded in the design, I2S0, I2S1 and I2S2. Different features between I2S/PCM controllers are as follows.

- Support four internal 32-bit wide and 32-location deep FIFOs for transmitting audio data for I2S0
- Support eight internal 32-bit wide and 32-location deep FIFOs, four for transmitting and four for receiving audio data for I2S1
- Support two internal 32-bit wide and 32-location deep FIFOs, one for transmitting and one for receiving audio data for I2S2
- Support 8 channels audio data transmitting in I2S mode for I2S0, 8 channels audio data transmitting or 8 channels audio data receiving for I2S1, 2 channels audio data transmitting and 2 channels audio data receiving for I2S2.

Common features for I2S0, I2S1 and I2S2 are as follows.

- 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 channels audio receiving in PCM mode
- 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 2 independent LRCK signals, one for receiving and one for transmitting audio data. Single LRCK can be used for transmitting and receiving data if the sample rate are the same
- Support configurable SCLK and LRCK polarity

10.2 Block Diagram

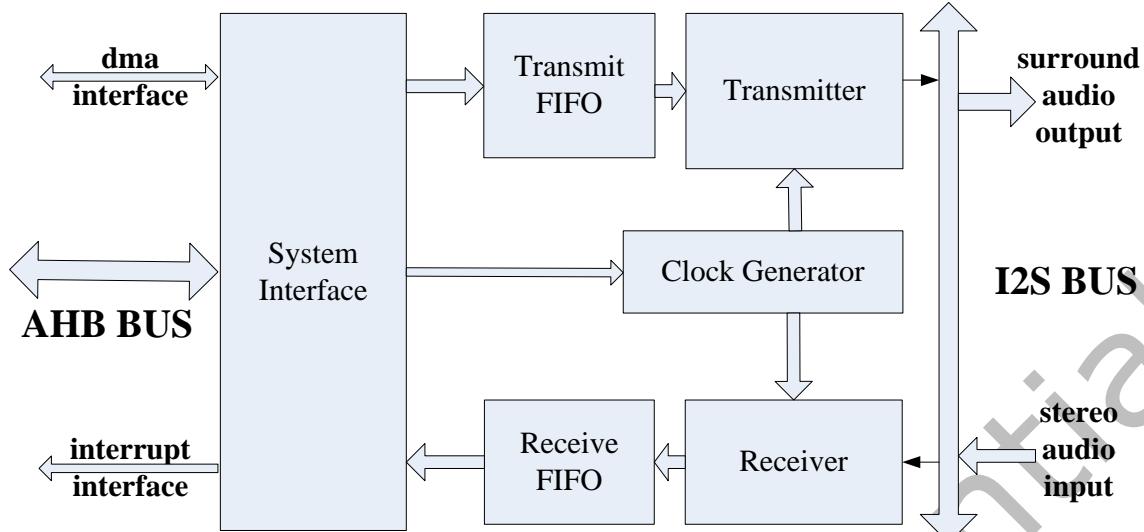


Fig. 10-1 I2S/PCM controller (8 channel) Block Diagram

System Interface

The system interface implements the AHB slave operation. It contains not only control registers of transmitter 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.

Transmitter

The Transmitter implements transmission operation. The transmitter can act as either master or slave, with I2S or PCM mode surround 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 FIFO

The Transmit FIFO is the buffer to store transmitted audio data. The size of the FIFO 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.

10.3 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.

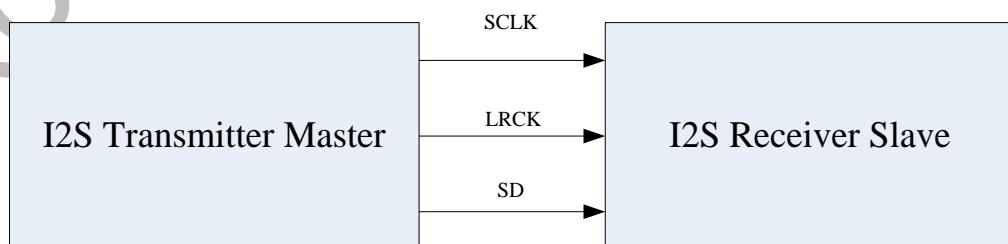


Fig. 10-2 I2S transmitter-master & receiver-slave condition

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.

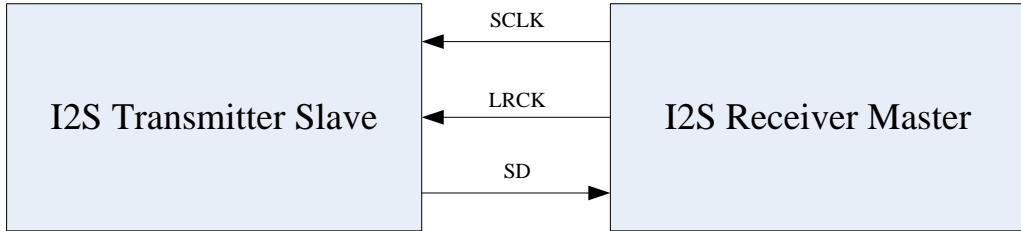


Fig. 10-3 I2S transmitter-slave& receiver-master condition

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 the receiver starts a transfer and sends 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.

10.3.1 i2s normal mode

This is the waveform of I2S normal mode. For LRCK (i2s_lrck_rx/i2s_lrck_tx) signal, it goes low to indicate left channel and high to right channel. For SD (i2s_sdo,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.

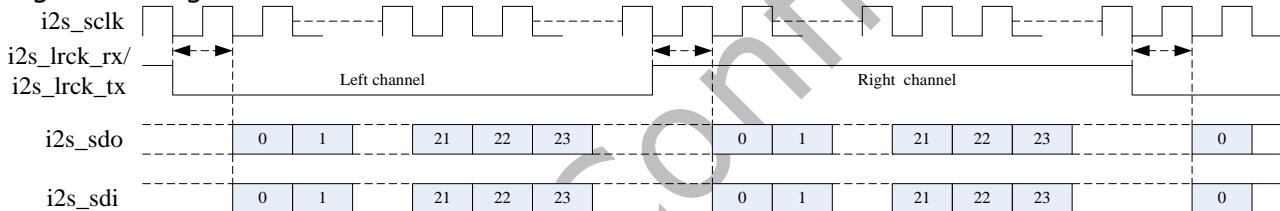


Fig. 10-4 I2S normal mode timing format

10.3.2 i2s left justified mode

This is the waveform of I2S left justified mode. For LRCK (i2s_lrck_rx / i2s_lrck_tx) signal, it goes high to indicate left channel and low to right channel. For SD (i2s_sdo, 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.

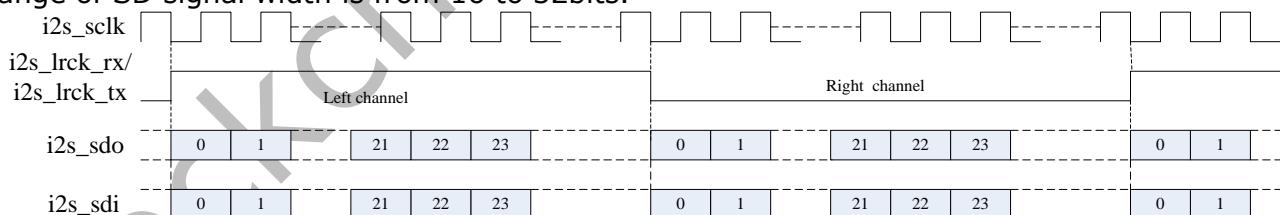


Fig. 10-5 I2S left justified mode timing format

10.3.3 i2s right justified mode

This is the waveform of I2S right justified mode. For LRCK (i2s_lrck_rx / i2s_lrck_tx) signal, it goes high to indicate left channel and low to right channel. For SD (i2s_sdo, 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.

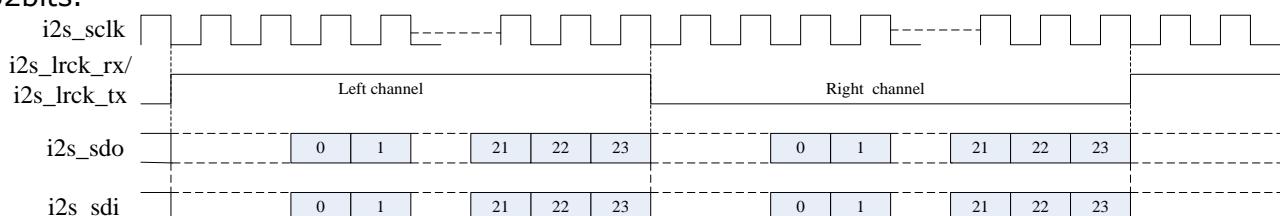


Fig. 10-6 I2S right justified mode timing format

10.3.4 PCM early mode

This is the waveform of PCM early mode. For LRCK (i2s_lrck_rx / i2s_lrck_tx) signal, it goes high to indicate the start of a group of audio channels. For SD (i2s_sdo, 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.

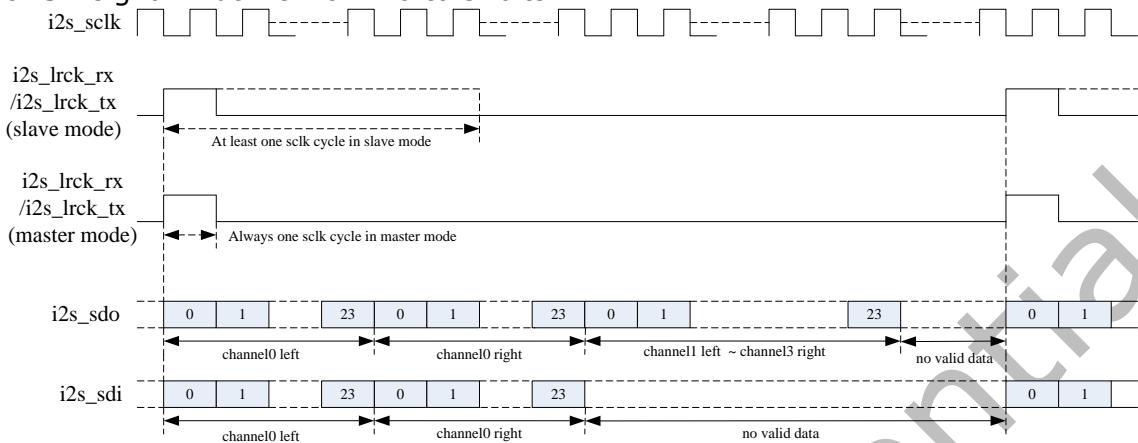


Fig. 10-7 PCM early mode timing format

10.3.5 PCM late1 mode

This is the waveform of PCM late1 mode. For LRCK (i2s_lrck_rx / i2s_lrck_tx) signal, it goes high to indicate the start of a group of audio channels. For SD (i2s_sdo, 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.

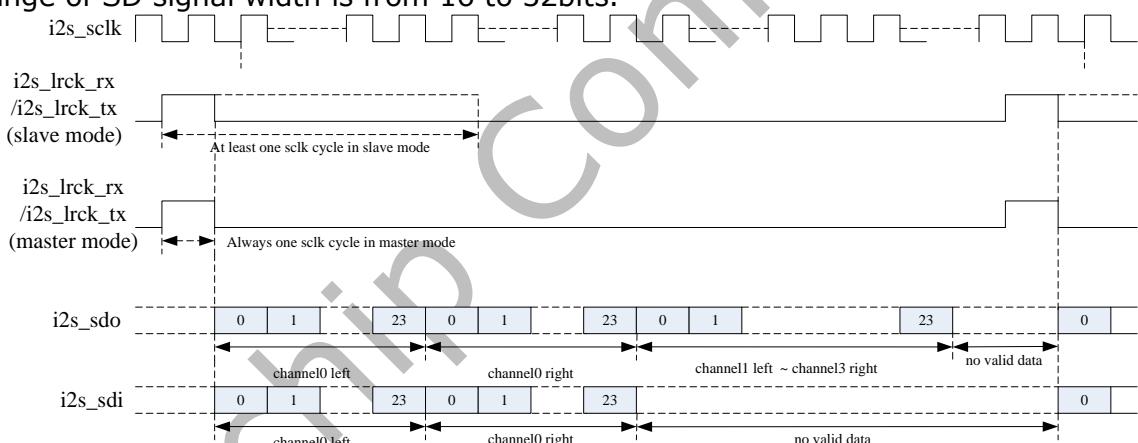


Fig. 10-8 PCM late1 mode timing format

10.3.6 PCM late2 mode

This is the waveform of PCM late2 mode. For LRCK (i2s_lrck_rx / i2s_lrck_tx) signal, it goes high to indicate the start of a group of audio channels. For SD (i2s_sdo, 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.

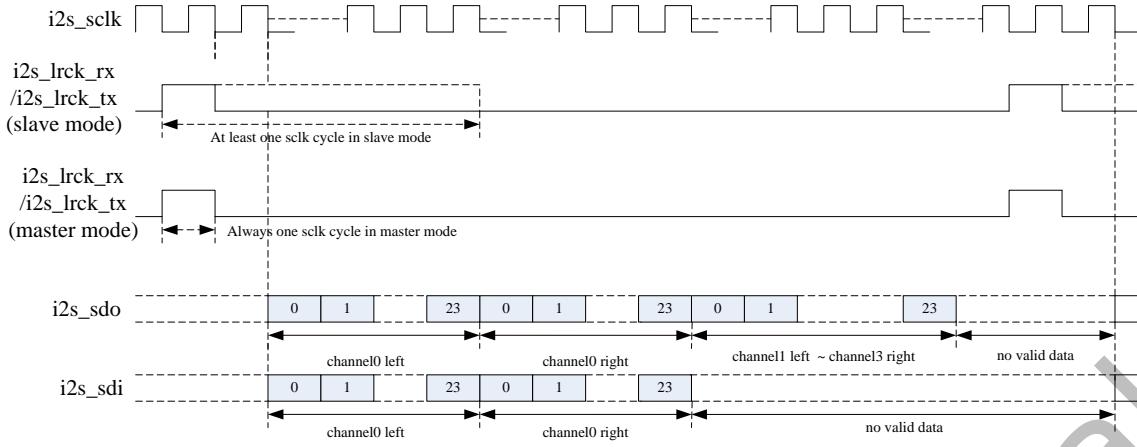


Fig. 10-9 PCM late2 mode timing format

10.3.7 PCM late3 mode

This is the waveform of PCM late3 mode. For LRCK (i2s_lrck_rx / i2s_lrck_tx) signal, it goes high to indicate the start of a group of audio channels. For SD (i2s_sdo, 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.

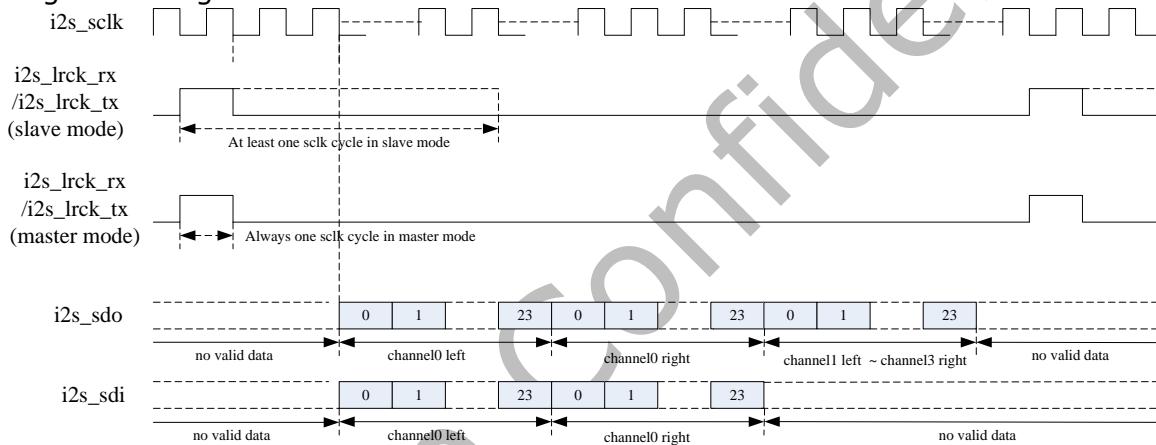


Fig. 10-10 PCM late3 mode timing format

10.4 Register Description

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

10.4.1 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_TXFIFOLR | 0x000c | W | 0x00000000 | TX FIFO level register |
| I2S_DMCR | 0x0010 | W | 0x001f0000 | DMA control register |
| I2S_INTCR | 0x0014 | W | 0x00000000 | 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 | Transmit FIFO Data Register |
| I2S_RXDR | 0x0028 | W | 0x00000000 | Receive FIFO Data Register |
| I2S_RXFIFOLR | 0x002c | W | 0x00000000 | RX FIFO level register |

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

10.4.2 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 | RCNT right justified counter (Can be written only when XFER[0] bit is 0.) Only valid in I2S Right justified format and slave tx mode is selected. Start to transmit data RCNT sclk cycles after left channel valid. |
| 16:15 | RW | 0x0 | TCSR TX Channel select register 2'b00:two channel 2'b01:four channel 2'b10:six channel 2'b11:eight channel |
| 14 | RW | 0x0 | 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. |
| 13 | RO | 0x0 | reserved |
| 12 | RW | 0x0 | SJM Store justified mode SJM Store justified mode (Can be written only when XFER[1] bit is 0.) 16bit~31bit DATA stored in 32 bits width fifo. This bit is invalid if VDW select 16bit data and HWT select 0, Because 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[0] bit is 0.) 0:MSB 1:LSB |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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 n:(n+1)bit 28:29bit 29:30bit 30:31bit 31:32bit |

I2S_RXCR

Address: Operational Base + offset (0x0004)
receive operation control register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:17 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 16:15 | RW | 0x0 | RCSR RX Channel select register 2'b00:two channel 2'b01:four channel 2'b10:six channel 2'b11:eight channel |
| 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 |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 5 | RW | 0x0 | TFS Transfer format select (Can be written only when XFER[1] bit is 0.) 0:i2s 1:pcm |
| 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 n:(n+1)bit 28:29bit 29:30bit 30:31bit 31:32bit |

I2S_CKR

Address: Operational Base + offset (0x0008)
clock generation register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 31:30 | RO | 0x0 | reserved |
| 29:28 | RW | 0x0 | TRCM Tx and Rx Common Use 2'b00/2'b11:tx_lrck/rx_lrck are used as synchronous signal for TX /RX respectively. 2'b01:only tx_lrck is used as synchronous signal for TX and RX. 2'b10:only rx_lrck is used as synchronous signal for TX and RX. |
| 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 |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 25 | RW | 0x0 | <p>RLP Receive Irck 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 Irck 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> |
| 23:16 | RW | 0x07 | <p>MDIV mclk divider (Can be written only when XFER[1] or XFER[0] bit is 0.)</p> <p>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; 2n,2n+1: Fmclk=(2n+2)*Ftxsclk; 60,61: Fmclk=62*Ftxsclk; 62,63: Fmclk=64*Ftxsclk; 252,253: Fmclk=254*Ftxsclk; 254,255: Fmclk=256*Ftxsclk;</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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 0~30:reserved 31: 32fs 32: 33fs 33: 34fs 34: 35fs n: (n+1)fs 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 0~30:reserved 31: 32fs 32: 33fs 33: 34fs 34: 35fs n: (n+1)fs 253: 254fs 254: 255fs 255: 256fs</p> |

I2S_TXFIFOLR

Address: Operational Base + offset (0x000c)

TX FIFO level register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:24 | RO | 0x0 | reserved |
| 23:18 | RO | 0x00 | <p>TFL3 Transmit FIFO3 Level Contains the number of valid data entries in the transmit FIFO3.</p> |
| 17:12 | RO | 0x00 | <p>TFL2 Transmit FIFO2 Level Contains the number of valid data entries in the transmit FIFO2.</p> |
| 11:6 | RO | 0x00 | <p>TFL1 Transmit FIFO1 Level Contains the number of valid data entries in the transmit FIFO1.</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 5:0 | RO | 0x00 | TFL0 Transmit FIFO0 Level Contains the number of valid data entries in the transmit FIFO0. |

I2S_DMCR

Address: Operational Base + offset (0x0010)

DMA control register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 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 (RXFIFO0 if RCSR=00;RXFIFO1 if RCSR=01,RXFIFO2 if RCSR=10,RXFIFO3 if RCSR=11)is equal to or above this field value + 1. |
| 15:9 | RO | 0x0 | reserved |
| 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 TCSR=00;TXFIFO1 if TCSR=01,TXFIFO2 if TCSR=10,TXFIFO3 if TCSR=11)is equal to or below this field value. |

I2S_INTCR

Address: Operational Base + offset (0x0014)

interrupt control register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|-------------|
| 31:25 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 24:20 | RW | 0x00 | RFT Receive FIFO Threshold When the number of receive FIFO entries (RXFIFO0 if RCSR=00; RXFIFO1 if RCSR=01, RXFIFO2 if RCSR=10, RXFIFO3 if RCSR=11) 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 TCSR=00; TXFIFO1 if TCSR=01, TXFIFO2 if TCSR=10, TXFIFO3 if TCSR=11) entries is less than or equal to this threshold, the transmit FIFO empty interrupt is triggered. |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:18 | RO | 0x0 | reserved |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:2 | RO | 0x0 | reserved |
| 1 | RW | 0x0 | RXS RX Transfer start bit 0:stop RX transfer. 1:start RX transfer |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 (0x0024)

Transmit FIFO Data Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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 (0x0028)

Receive FIFO Data Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RO | 0x00000000 | RXDR Receive FIFO Data Register When the register is read, data in the receive FIFO is accessed. |

I2S_RXFIFOLR

Address: Operational Base + offset (0x002c)

RX FIFO level register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:24 | RO | 0x0 | reserved |
| 23:18 | RO | 0x00 | RFL3 Receive FIFO3 Level Contains the number of valid data entries in the receive FIFO3. |
| 17:12 | RO | 0x00 | RFL2 Receive FIFO2 Level Contains the number of valid data entries in the receive FIFO2. |
| 11:6 | RU | 0x00 | RFL1 Receive FIFO1 Level Contains the number of valid data entries in the receive FIFO1. |
| 5:0 | RO | 0x00 | RFL0 Receive FIFO0 Level Contains the number of valid data entries in the receive FIFO0. |

10.5 Interface description

Table 10-1 I2S Interface Description

| Module Pin | Direction | Pad Name | IOMUX Setting |
|--------------------|------------------|-----------------------------|-------------------------------|
| Interface for i2s1 | | | |
| i2s1_mclk | I/O | IO_I2Smclk_GPIO0b0 | GRF_GPIO0B_IOMUX[1:0]=2'b01 |
| i2s1_sclk | I/O | IO_I2Ssclk_SPIclk_GPIO0b1 | GRF_GPIO0B_IOMUX[3:2]=2'b01 |
| i2s1_lrck_rx | I/O | IO_I2Slrckrx_SPItxd_GPIO0b3 | GRF_GPIO0B_IOMUX[7:6]=2'b01 |
| i2s1_lrck_tx | I/O | IO_I2Slrcktx_GPIOb4 | GRF_GPIO0B_IOMUX[9:8]=2'b01 |
| i2s1_sdo0 | O | IO_I2Ssdo_SPIrxd_GPIO0b5 | GRF_GPIO0B_IOMUX[11:10]=2'b01 |
| i2s1_sdo1 | O | IO_SDIOd1_I2Ssdo1_GPIO1a2 | GRF_GPIO1A_IOMUX[5:4]=2'b10 |
| i2s1_sdo2 | O | IO_SDIOd2_I2Ssdo2_GPIO1a4 | GRF_GPIO1A_IOMUX[9:8]=2'b10 |
| i2s1_sdo3 | O | IO_SDIOd3_I2Ssdo3_GPIO1a5 | GRF_GPIO1A_IOMUX[11:10]=2'b10 |
| i2s1_sdi0 | I | IO_I2Ssdi_SPIcsn0_GPIO0b6 | GRF_GPIO0B_IOMUX[13:12]=2'b01 |
| i2s1_sdi1 | I | IO_SDIOd1_I2Ssdo1_GPIO1a2 | GRF_GPIO1A_IOMUX[5:4]=2'b10 |
| i2s1_sdi2 | I | IO_SDIOd2_I2Ssdo2_GPIO1a4 | GRF_GPIO1A_IOMUX[9:8]=2'b10 |
| i2s1_sdi3 | I | IO_SDIOd3_I2Ssdo3_GPIO1a5 | GRF_GPIO1A_IOMUX[11:10]=2'b10 |
| Interface for i2s2 | | | |

| Module Pin | Direction | Pad Name | IOMUX Setting |
|-------------------|------------------|-----------------------|-----------------------------|
| i2s2_sclk | I/O | IO_PCMclk_GPIO3b3 | GRF_GPIO3B_IOMUX[7:6]=2'b01 |
| i2s2_lrck | I/O | IO_PCMsync_GPIO3b4 | GRF_GPIO3B_IOMUX[9:8]=2'b01 |
| i2s2_sdi | I | IO_PWM0_PCMrx_GPIO0d2 | GRF_GPIO0D_IOMUX[5:4]=2'b10 |
| i2s2_sdo | O | IO_PWM1_PCMtx_GPIO0d3 | GRF_GPIO0D_IOMUX[7:6]=2'b10 |

Notes: I=input, O=output, I/O=input/output, bidirectional

The i2s1_sdix(x=1,2,3) and i2s1_sdox(x=1,2,3) signals shares the same IO, the direction is configured by setting GRF_CON_IOMUX[14:12]. Each bit of GRF_CON_IOMUX [14:12] controls the direction of IO_SDIOd3_I2Ssdio3_GPIO1a5, IO_SDIOd2_I2Ssdio2_GPIO1a4 and IO_SDIOd1_I2Ssdio1_GPIO1a2 respectively with high level meaning output.

I2s2_lrck is connected to i2s2_lrck_rx and i2s2_lrck_tx at the same time, so the sample rate of transmitting and receiving must be the same.

The I2S1 is also connected to the ACODEC which supports master and slave mode. When the ACODEC acts as a master, the signal i2s1_lrck_tx_in which connected to I2S1 can be selected from ACODEC or external IO by setting GRF_CON_IOMUX[6].

Table 10-2 Interface Between I2S1 and ACODEC

| Module Pin | Direction | Module Pin | Direction |
|-------------------|------------------|-------------------|------------------|
| i2s1_mclk | O | pin_mclk | I |
| i2s1_sclk_out | O | pin_sck_i | I |
| i2s1_sclk_in | I | pin_sck_o | O |
| i2s1_lrck_tx_out | O | pin_dac_ws_i | I |
| i2s1_lrck_tx_in | I | pin_dac_ws_o | O |
| i2s1_sdo0 | O | pin_dac_sd_i | I |

The I2S0 module is connected to the audio interface of HDMI, which supports 8 channels audio data transmitting.

Table 10-3 I2S Interface Between I2S2 and HDMI

| Module Pin | Direction | Module Pin | Direction |
|-------------------|------------------|-------------------|------------------|
| i2s0_sclk_out | O | ii2sclk | I |
| i2s0_tx_lrck_out | O | ii2slrck | I |
| i2s0_sdo[3:0] | O | ii2sdata[3:0] | I |

10.6 Application Notes

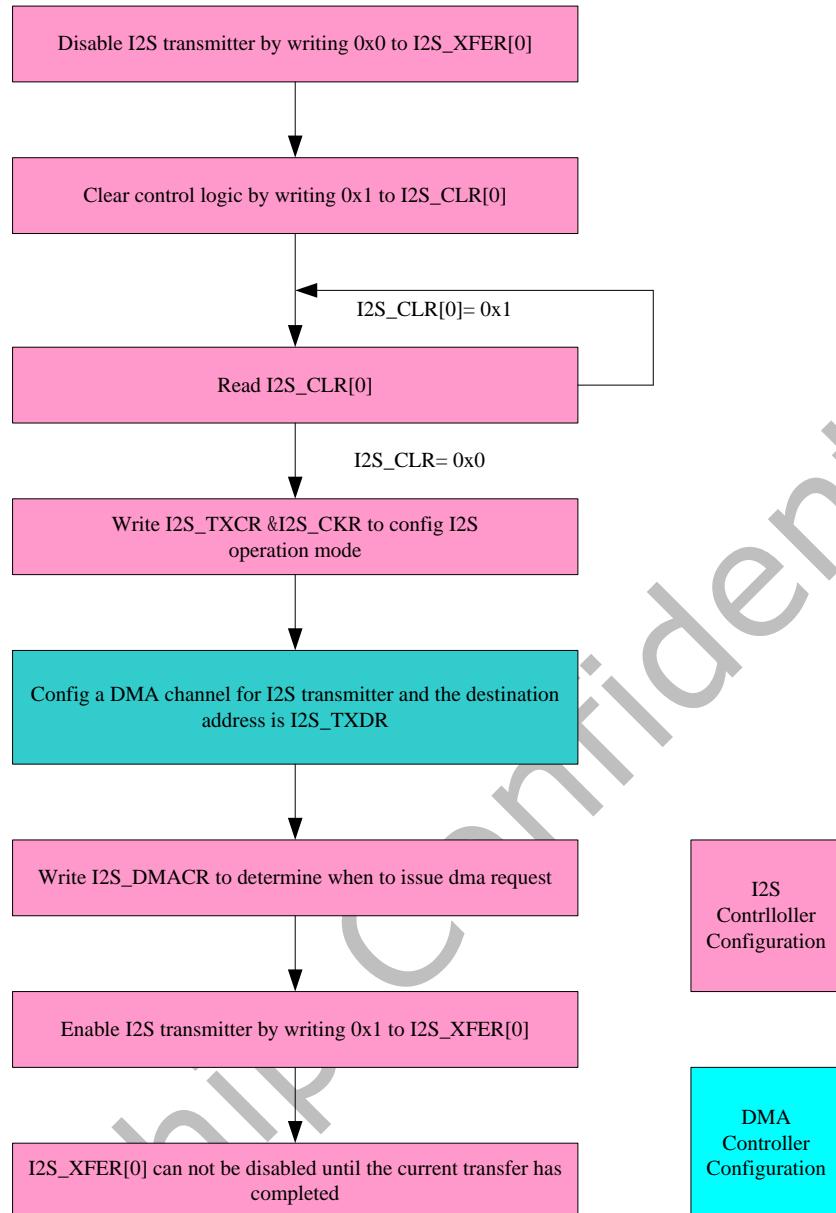


Fig. 10-11 I2S/PCM controller transmit operation flow chart

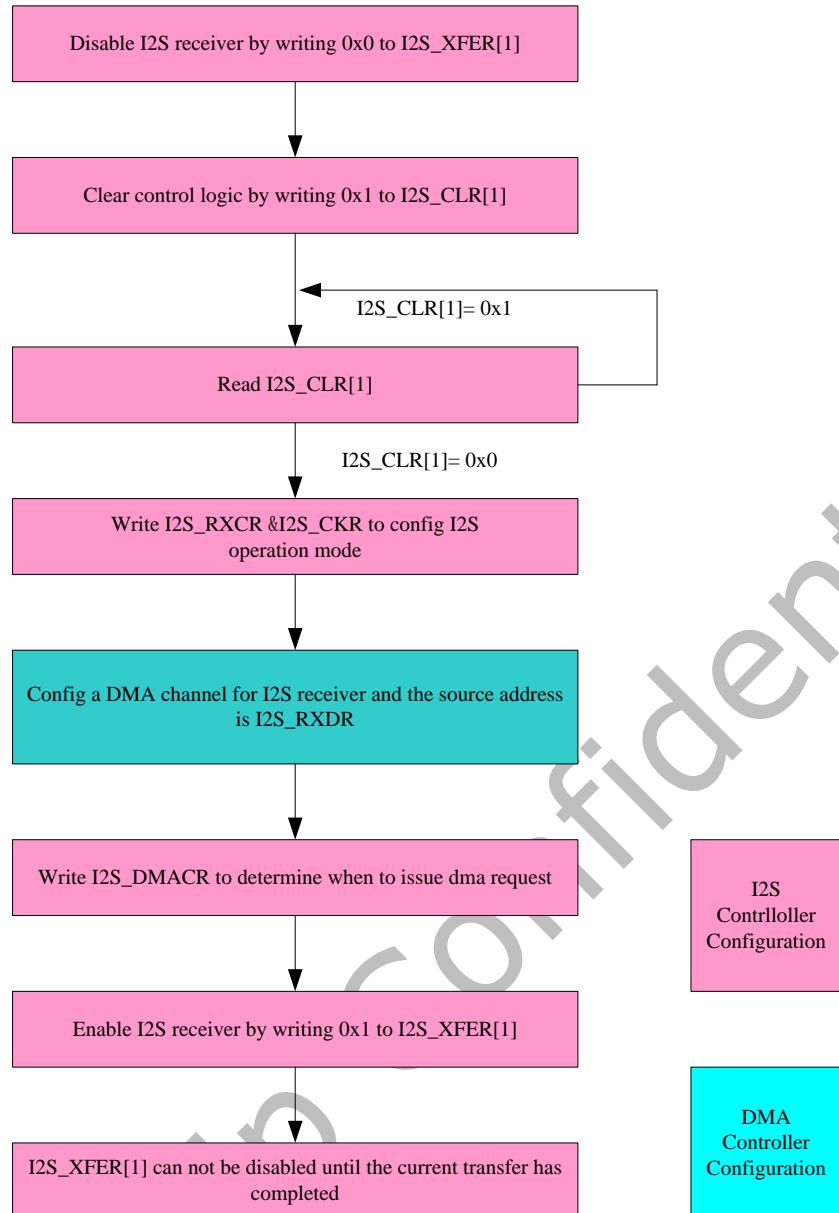


Fig. 10-12 I2S/PCM controller receive operation flow chart

Chapter 11 Serial Peripheral Interface (SPI)

11.1 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 Micro wire 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

11.2 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

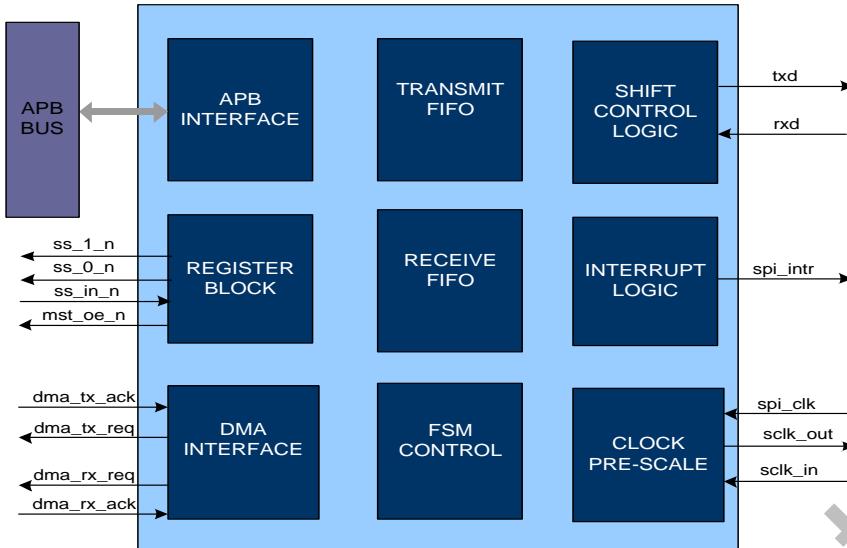


Fig. 10-13 SPI Controller Block diagram

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 32 bits and 8 or 16 bits when reading or writing internal FIFO if data frame size(SPI_CTRL0[1:0]) is set to 8 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 APB 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.

11.3 Function Description

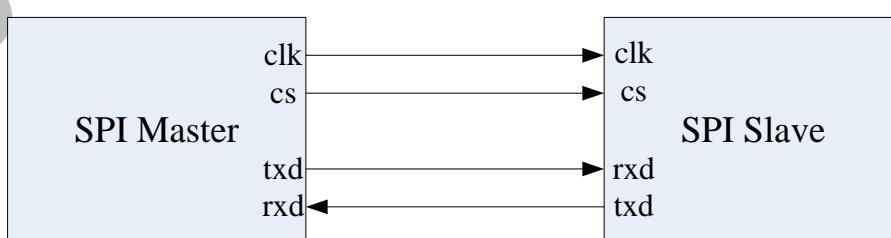


Fig. 10-14 SPI Master and Slave Interconnection

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 SCPOL = 1.

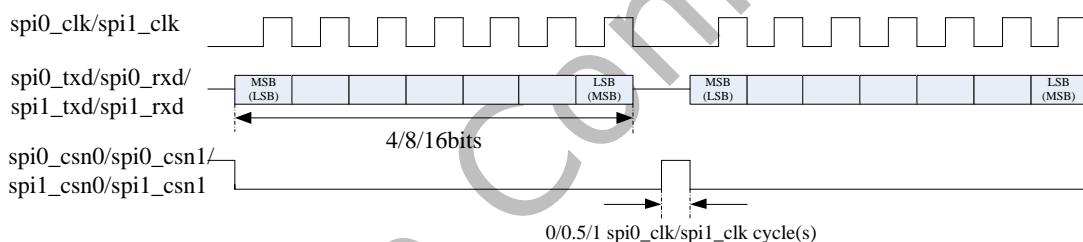


Fig. 10-15 SPI Format (SCPH=0 SCPOL=0)

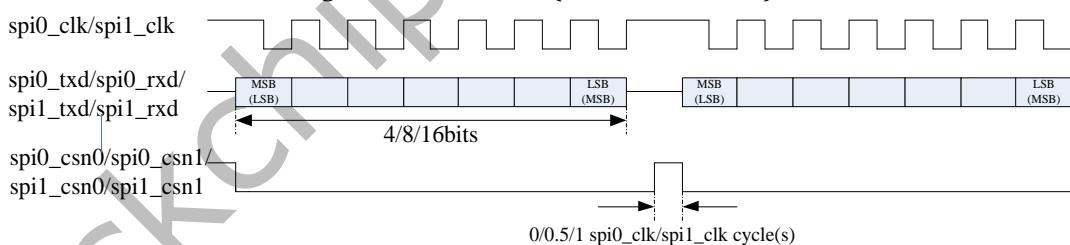


Fig. 10-16 SPI Format (SCPH=0 SCPOL=1)

When the configuration parameter 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 SCPH = 1.

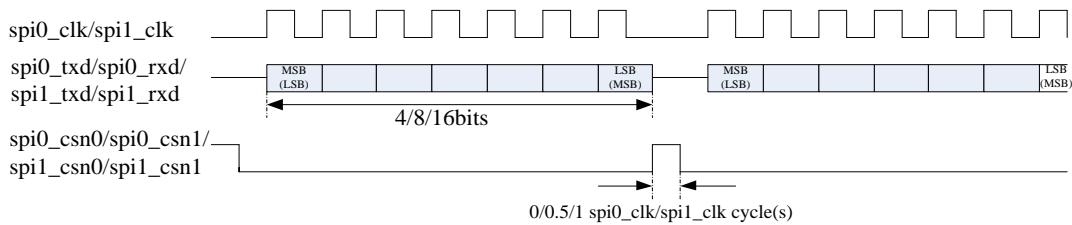


Fig. 10-17 SPI Format (SCPH=1 SCPOL=0)

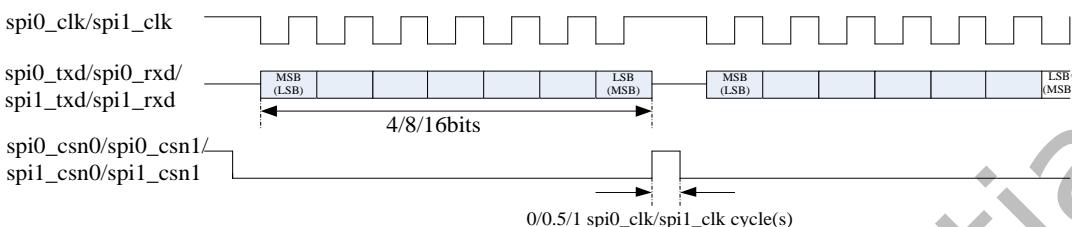


Fig. 10-18 SPI Format (SCPH=1 SCPOL=1)

11.4 Register Description

11.4.1 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 | 0x0048 | W | 0x00000000 | Transmit FIFO Data |
| SPI_RXDR | 0x004c | W | 0x00000000 | Receive FIFO Data |

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

11.4.2 Detail Register Description

SPI_CTRLR0

Address: Operational Base + offset (0x0000)

Control Register 0

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|-------------|
| 31:22 | RO | 0x0 | reserved |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 11 | RW | 0x0 | <p>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</p> |
| 10 | RW | 0x0 | <p>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.</p> |
| 9:8 | RW | 0x0 | <p>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</p> |
| 7 | RW | 0x0 | <p>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</p> |
| 6 | RW | 0x0 | <p>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</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 5:2 | RW | 0x0 | CFS Control Frame Size Selects the length of the control word for the Microwire frame format. 4'b0000~0010: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

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|-------------|
| 31:1 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 0 | RW | 0x0 | <p>ENR SPI Enable 1'b1: Enable all SPI operations. 1'b0: Disable all SPI operations Transmit and receive FIFO buffers are cleared when the device is disabled.</p> |

SPI_SER

Address: Operational Base + offset (0x000c)

Slave Enable Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:2 | RO | 0x0 | reserved |
| 1 | RW | 0x0 | <p>SER1 Slave 1 Select Enable 1'b1: Enable chip select 1 1'b0: Disable chip select 1 This register is valid only when SPI is configured as a master device.</p> |
| 0 | RW | 0x0 | <p>SERO Slave Select Enable 1'b1: Enable chip select 0 1'b0: Disable chip select 0 This register is valid only when SPI is configured as a master device.</p> |

SPI_BAUDR

Address: Operational Base + offset (0x0010)

Baud Rate Select

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:16 | RO | 0x0 | reserved |
| 15:0 | RW | 0x0000 | <p>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 $SCKDV = 2$ $F_{sclk_out} = 3.6864/2 = 1.8432\text{MHz}$</p> |

SPI_TXFTLR

Address: Operational Base + offset (0x0014)

Transmit FIFO Threshold Level

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:5 | RO | 0x0 | reserved |

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

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

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

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

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

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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 (0x0048)

Transmit FIFO Data

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:16 | RO | 0x0 | reserved |
| 15:0 | WO | 0x0000 | TXDR Transimt FIFO Data Register. When it is written to, data are moved into the transmit FIFO. |

SPI_RXDR

Address: Operational Base + offset (0x004c)

Receive FIFO Data

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. |

11.5 Interface Description

Table 10-4 1SPI interface description

| Module Pin | Direction | Pad Name | IOMUX Setting |
|-------------------|------------------|-----------------------------|-------------------------------|
| spi_clk | I/O | IO_I2Ssclk_SPIclk_GPIO0b1 | GRF_GPIO0B_IOMUX[3:2]=2'b10 |
| spi_rxd | I | IO_I2Ssdo_SPIrxd_GPIO0b5 | GRF_GPIO0B_IOMUX[11:10]=2'b10 |
| spi_txd | O | IO_I2Slrckrx_SPItxd_GPIO0b3 | GRF_GPIO0B_IOMUX[7:6]=2'b10 |
| spi_csn0 | I/O | IO_I2Ssdi_SPIcsn0_GPIO0b6 | GRF_GPIO0B_IOMUX[13:12]=2'b10 |
| spi_csn1 | O | IO_SPIcsn1_PWM12_GPIO1b4 | GRF_GPIO1B_IOMUX[9:8]=2'b01 |
| spi1_clk | I/O | IO_NANDcs1_SPI1clk_GPIO0c7 | GRF_GPIO0C_IOMUX[15:14]=2'b10 |
| spi1_rxd | I | IO_NANDale_SPI1rxd_GPIO2a0 | GRF_GPIO2A_IOMUX[1:0]=2'b10 |
| spi1_txd | O | IO_NANDcle_SPI1txd_GPIO2a1 | GRF_GPIO2A_IOMUX[3:2]=2'b10 |
| spi1_csn0 | I/O | IO_NANDwrn_SPI1csn0_GPIO2a2 | GRF_GPIO2A_IOMUX[5:4]=2'b10 |
| spi1_csn1 | O | IO_NANDrdn_SPI1csn1_GPIO2a3 | GRF_GPIO2A_IOMUX[7:6]=2'b10 |

Notes: I=input, O=output, I/O=input/output, bidirectional. spi_csn1 can only be used in master mode

11.6 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 $F_{spi_clk} \geq 2 \times (\text{maximum } F_{sclk_out})$ When SPI Controller works as slave, the $F_{spi_clk} \geq 6 \times (\text{maximum } F_{sclk_in})$ **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.

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.

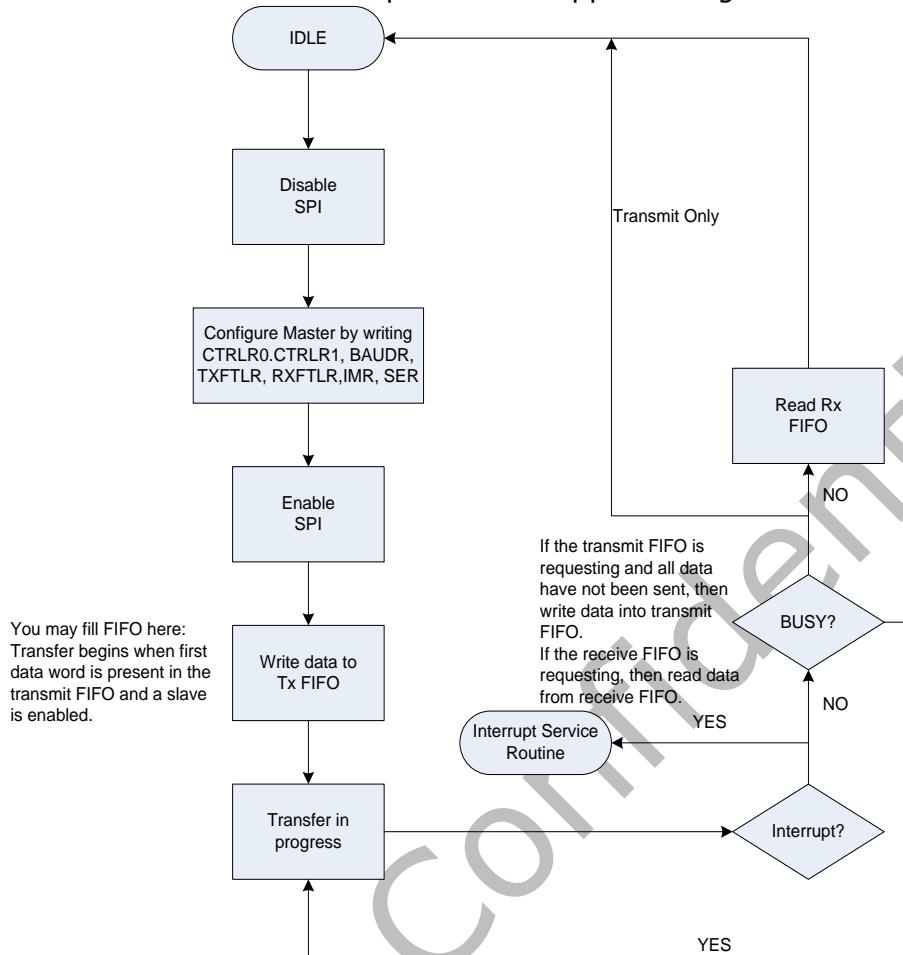


Fig. 10-19 SPI Master transfer flow diagram

Chapter 12 SPDIF transmitter

12.1 Overview

The SPDIF transmitter is a self-clocking, serial and unidirectional interface for the interconnection of digital audio equipment in consumer and professional applications which uses linear PCM coded audio samples.

When used in professional application, the interface is primarily intended to carry monophonic or stereophonic programmes at a 48 kHz sampling frequency 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 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 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 maximum sample frequency can be up to 192 kHz for the non-linear PCM mode.

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

12.2 Block Diagram

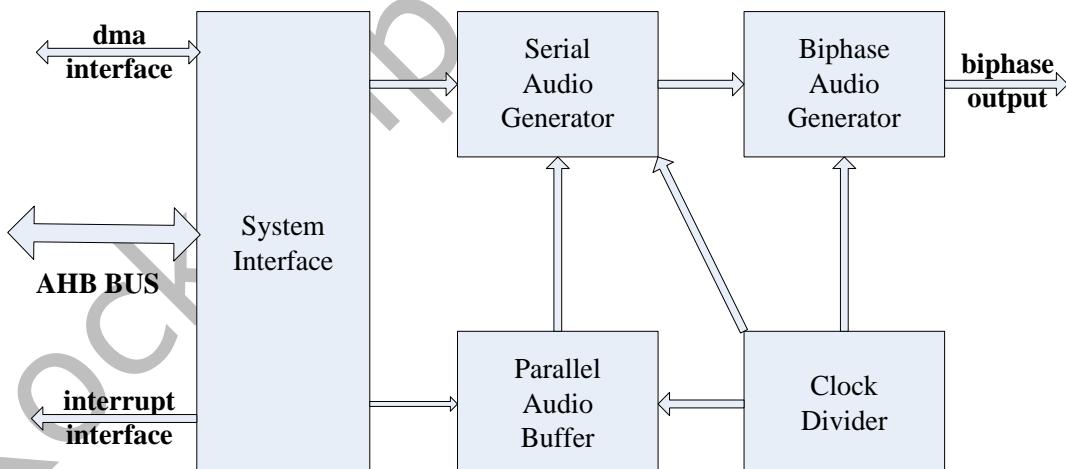


Fig. 10-20 SPDIF transmitter Block Diagram

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. 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.

12.3 Function description

12.3.1 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.

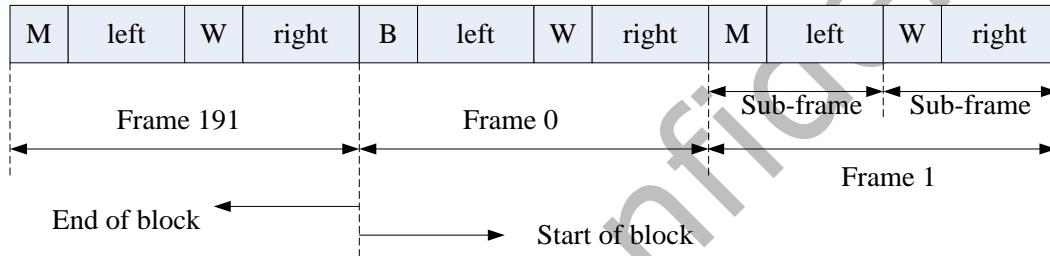


Fig. 10-21 SPDIF Frame Format

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).

12.3.2 Sub-frame Format

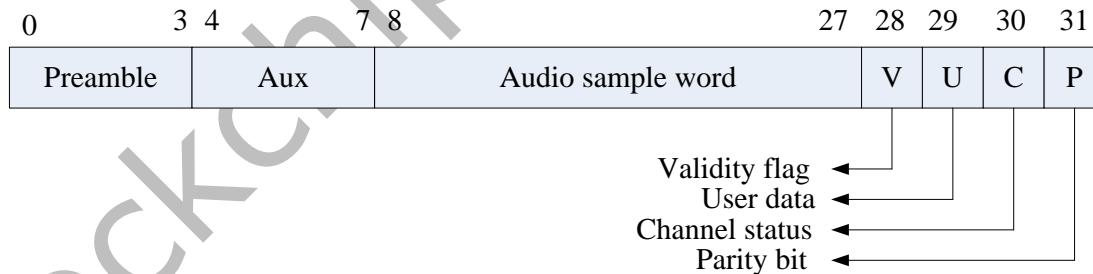


Fig. 10-22 SPDIF Sub-frame Format

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.

12.3.3 Channel Coding

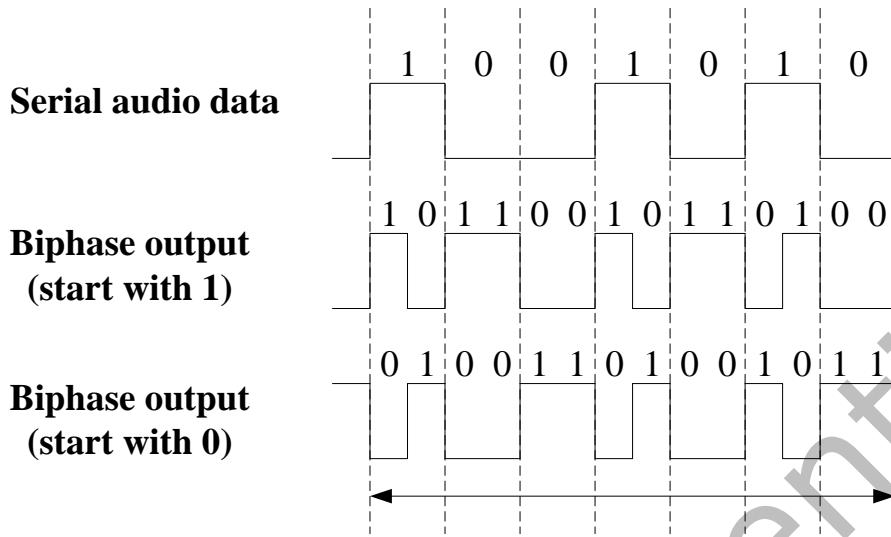


Fig. 10-23 SPDIF Channel Coding

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 biphasic-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'.

12.3.4 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 biphasic-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.

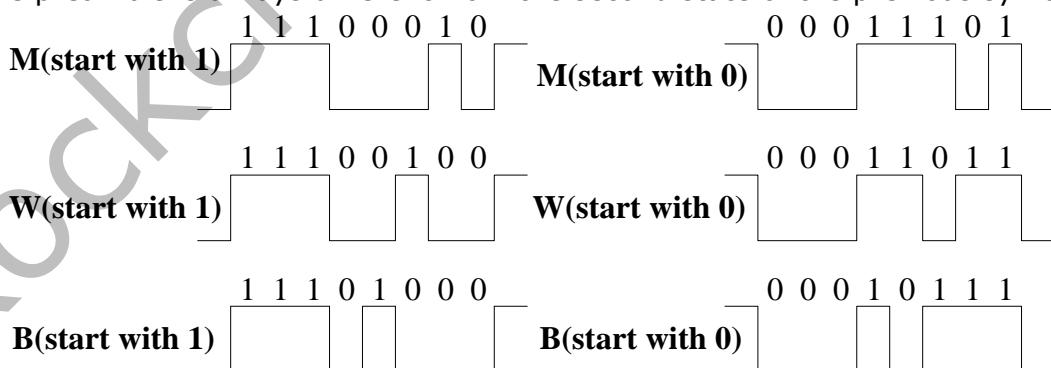


Fig. 10-24 SPDIF Preamble

Like biphasic code, these preambles are dc free and provide clock recovery. They differ in at least two states from any valid biphasic sequence.

12.3.5 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 burst payload 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 burst payload, 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 timeslot 27 and the LSB is placed into time slot 12.

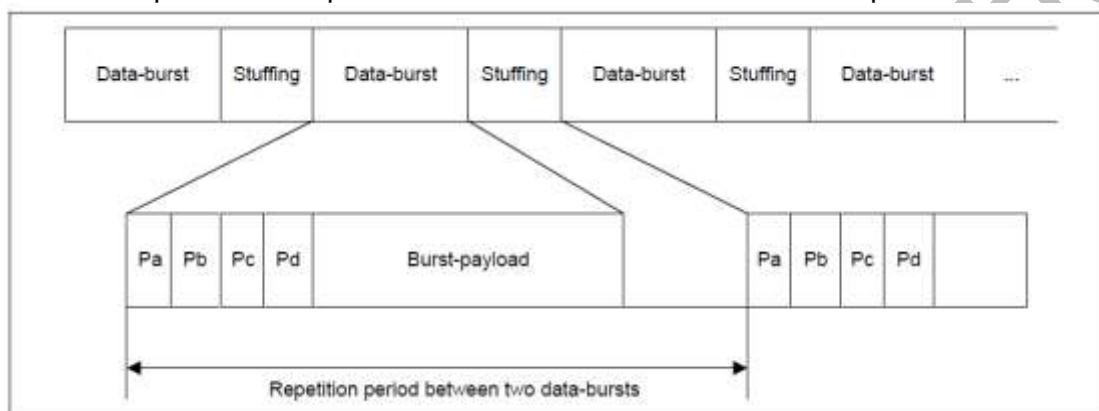


Fig. 10-25 Format of Data-burst

12.4 Register description

12.4.1 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: **S**-ize: **B**- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**-WORD (32 bits) access

12.4.2 Detail Register Description

SPDIF_CFGR

Address: Operational Base + offset (0x0000)

Transfer Configuration Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:24 | RO | 0x0 | reserved |
| 23:16 | RW | 0x00 | MCD mclk divider Fmclk/Fsdo This parameter can be calculated by Fmclk/(Fs*128). Fs=the sample frequency be wanted |
| 15:9 | RO | 0x0 | reserved |
| 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 |
| 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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:18 | RO | 0x0 | reserved |
| 17 | W1C | 0x0 | UDTIC User Data Interrupt Clear Write '1' to clear the user data interrupt. |
| 16 | W1C | 0x0 | BTIC 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 |

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|---|
| 31:5 | RO | 0x0 | reserved |
| 4 | RW | 0x0 | SDBEIS Sample Date Buffer empty interrupt status 0: inactive 1: active |
| 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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RW | 0x00000000 | SMPDR Sample Data Register Sample Data Register |

SPDIF_VLDFRn

Address: Operational Base + offset (0x0060)

Validity Flag Register n

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 (0x00d0)

Channel Burst Info Register

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 bitstream 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 |

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

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 31:16 | RO | 0x0 | reserved |
| 15:0 | RW | 0x0000 | <p>REPETITION Repetition This define the repetition period when the channel conveys non-linear PCM</p> |

SPDIF_BURTSINFO_SHD

Address: Operational Base + offset (0x0108)

Shadow Channel Burst Info Register

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

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 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_REPEAT_SHD

Address: Operational Base + offset (0x010c)

Shadow Channel Repetition Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|-------------|
| 31:16 | RO | 0x0 | reserved |

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

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

12.5 Interface description

Table 10-5 SPDIF Interface Description

| Module Pin | Direction | Pad Name | IOMUX Setting |
|---------------|-----------|-------------------------------------|-------------------------------|
| spdif_8ch_sdo | O | IO_SPDIFtx_GPIO3d3 | GRF_GPIO3D_IOMUX[7:6]=2'b01 |
| spdif_8ch_sdo | O | IO_TESTCLKout1_SPDIF1tx_GP IO3d7 | GRF_GPIO3D_IOMUX[15:14]=2'b10 |

The output of SPDIF module which signals as spdif_8ch_sdo is also connected to the audio interface of HDMI.

Table 10-6 Interface Between SPDIF And HDMI

| Module Pin | Direction | Module Pin | Direction |
|----------------|-----------|------------|-----------|
| mclk_spdif_8ch | O | ispdifclk | I |
| spdif_8ch_sdo | O | ispdifdata | I |

12.6 Application Notes

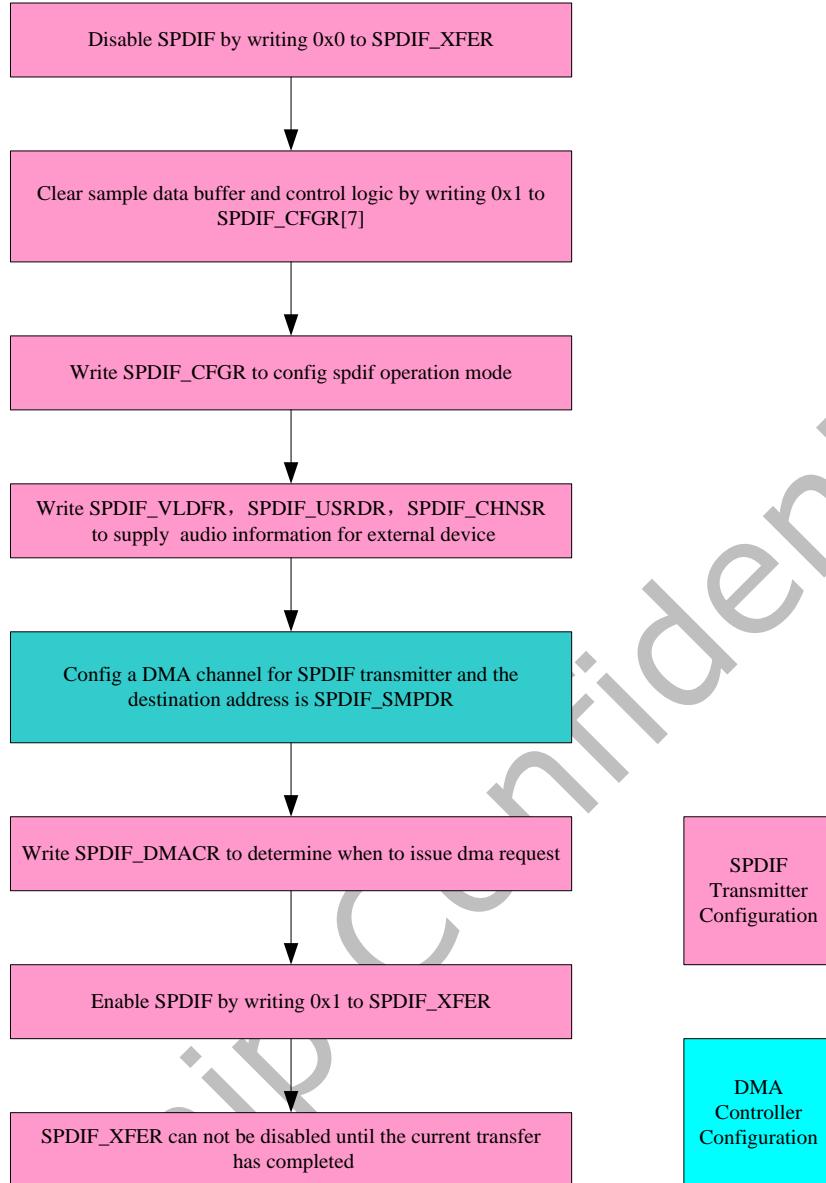


Fig. 10-26 SPDIF transmitter operation flow chart

12.6.1 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 termination interrupt is asserted. The new value will take effect immediately.

12.6.2 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.

12.6.3 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 termination 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 calculate the new repetition value if the definition of the repetition is different for some audio formats such as AC-3.

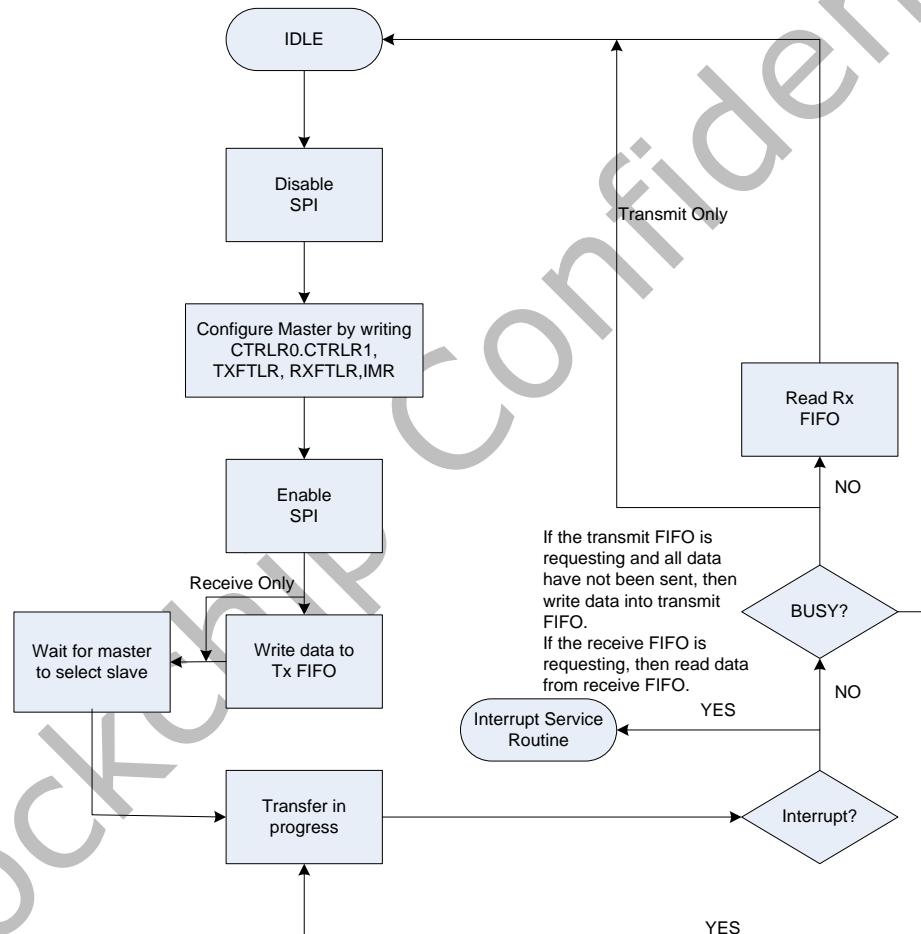


Fig. 10-27 SPI Slave transfer flow diagram

Chapter 13 TSP(Transport Stream Processing Module)

13.1 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 1 TS input channels
- 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 2 TS sources: demodulators and local memory
- Supports 1 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
- Support DMA LLP transfer
- 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

13.2 Block Diagram

The TSP comprises of following components:

- AMBA AHB slave interface
- Register block
- PTI
- DMAC
- TS Out Interface

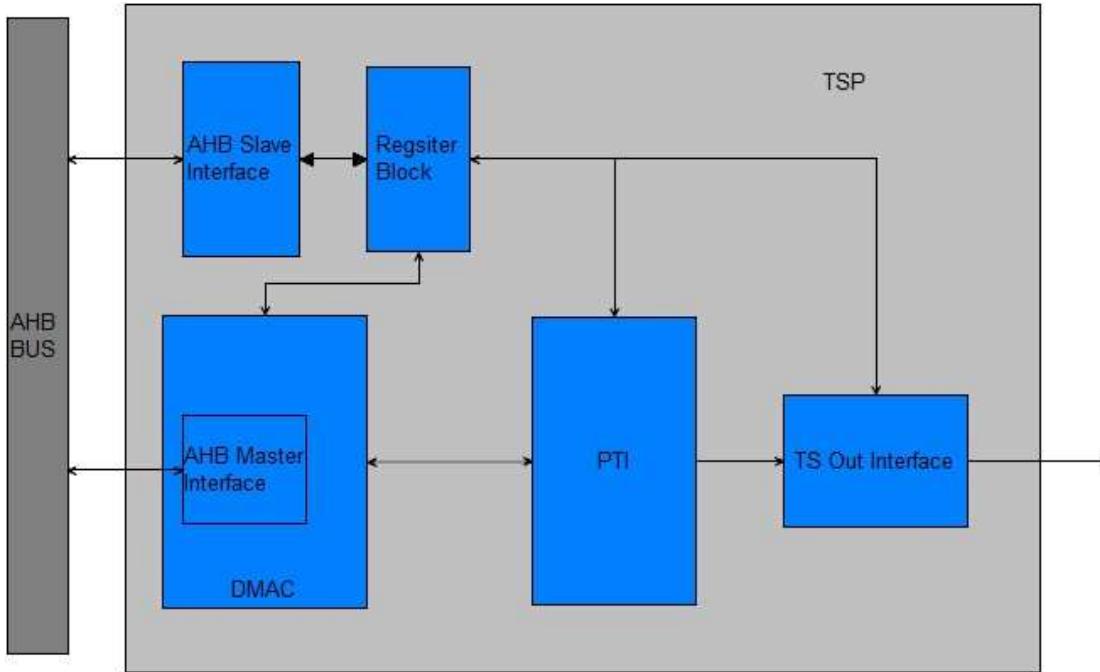


Fig. 1-28 TSP architecture

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.

13.3 Function Description

13.3.1 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.

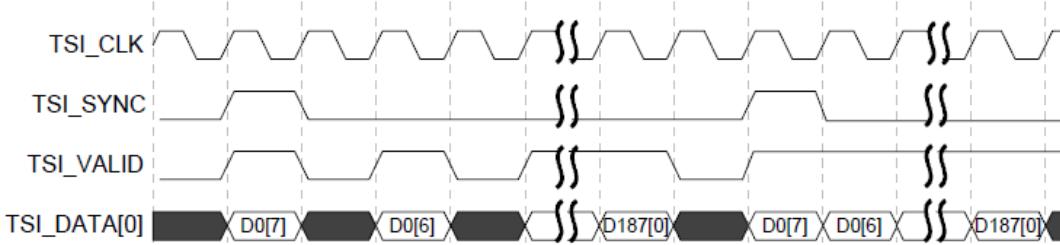


Fig. 1-29 Sync/Valid Serial Mode with Msb-Lsb Bit Ordering

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.

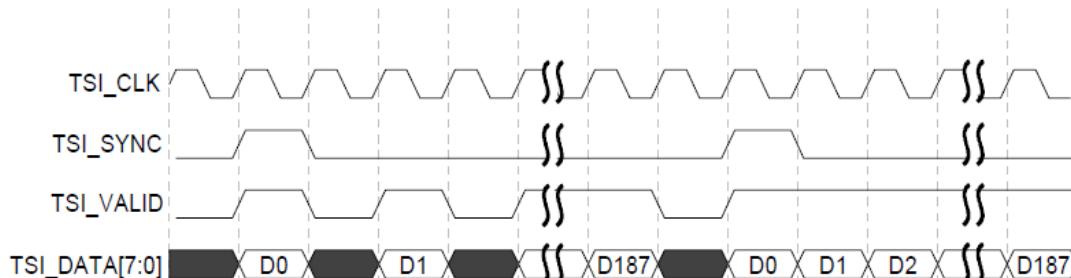


Fig. 1-30 Sync/valid Parallel Mode

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.

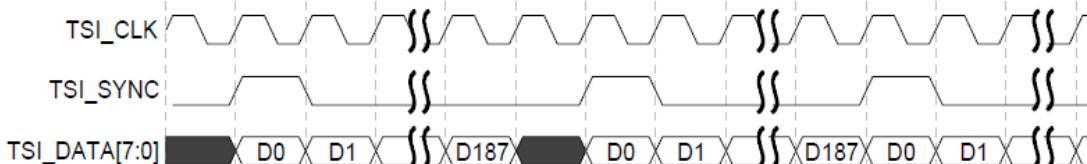


Fig. 1-31 Sync/Burst Parallel Mode

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.

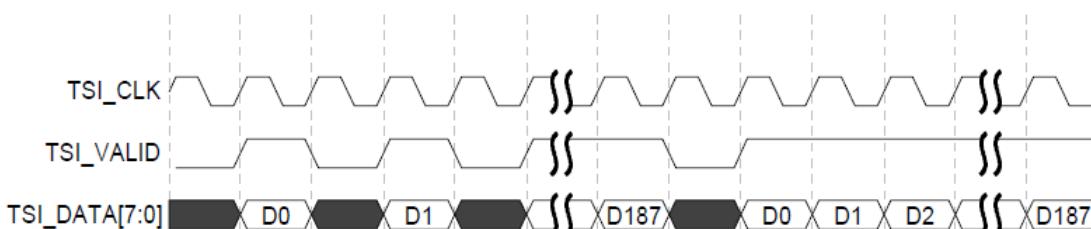


Fig. 1-32 Nosync/Valid Parallel Mode

When active high, TSI_VALID implies a valid byte of a TS packet.

13.3.2 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.

13.3.3 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.

13.4 Register Description

13.4.1 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_BASE_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_PVR_TOP_ADDR | 0x001c | W | 0x00000000 | |
| TSP_PVR_WRITE_ADDR | 0x0020 | W | 0x00000000 | |
| TSP_PTIX_CTRL | 0x0100 | W | 0x00000000 | PTI Channel Control Register |
| TSP_PTIX_LLPCFG | 0x0104 | W | 0x00000000 | LLP DMA Control Register |
| TSP_PTIX_LLPPBASE | 0x0108 | W | 0x00000000 | LLP Descriptor BASE Address |
| TSP_PTIX_LLPPWRITE | 0x010c | W | 0x00000000 | LLP DMA Writing Software Descriptor Counter |

| Name | Offset | Size | Reset Value | Description |
|-----------------------|--------|------|-------------|---|
| TSP_PTIx_LL_P_R | 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_CTR_L | 0x013c | W | 0x00000000 | PID PCR Control Register |
| 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_FLA_G | 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_ENA0 | 0x01b8 | W | 0x00000000 | PTI Destination Interrupt Enable Register |
| TSP_PTIx_DST_ENA1 | 0x01bc | W | 0x00000000 | PTI Destination Interrupt Enable Register |

| Name | Offset | Size | Reset Value | Description |
|------------------------|--------|------|-------------|---|
| 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 |
| 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

13.4.2 Detail Register Description

TSP_GCFG

Address: Operational Base + offset (0x0000)

Global Configuration Register

| Bit | Attr | Reset Value | Description |
|------|------|-------------|-------------|
| 31:7 | RO | 0x0 | reserved |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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: Reserverd |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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 takes 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RW | 0x00000000 | len Transaction Length Transaction Length |

TSP_PVR_BASE_ADDR

Address: Operational Base + offset (0x000c)

PVR DMA transaction starting address

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:3 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 2 | W1C | 0x0 | pvr_update_flag pvr address pageover flag When write_addr >= (base + top_addr/2), or write addr >= top_addr, the pvr_update_flag will assert HIGH. The application can write 1 to this bit to clear it. |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:3 | RO | 0x0 | reserved |
| 2 | RW | 0x0 | pvr_update_ena 1: pvr_update interrupt enable 0: pvr_update interrupt disable |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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 ; |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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_PVR_TOP_ADDR

Address: Operational Base + offset (0x001c)

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:0 | RW | 0x00000000 | pvr_top_addr |

TSP_PVR_WRITE_ADDR

Address: Operational Base + offset (0x0020)

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RO | 0x00000000 | pvr_write_addr The core will update this register to show the PVR write addr |

TSP_PTIX_CTRL

Address: Operational Base + offset (0x0100)

PTI Channel Control Register

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:10 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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> |

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

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

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

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

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

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

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

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

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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 |
| 28 | W1C | 0x0 | pid60_error PID60 Error Interrupt Status |
| 27 | W1C | 0x0 | pid59_error PID59 Error Interrupt Status |

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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; |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:8 | RO | 0x0 | reserved |
| 7 | RW | 0x0 | pcr7_done_ena pcr7 done interrupt enable 1: enabled; 0: disabled; |
| 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; |

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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; |

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 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 | R/WSC | 0x0 | clear PID Channel Clear Write 1 to clear PID channel. This bit will be set to 0 if the channel is clear. |
| 0 | R/WSC | 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

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

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

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

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

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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 |
| 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. |

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

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

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

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

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

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

Notes: I=input, O=output, I/O=input/output, bidirectional

13.5 Application Notes

13.5.1 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.

13.5.2 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.

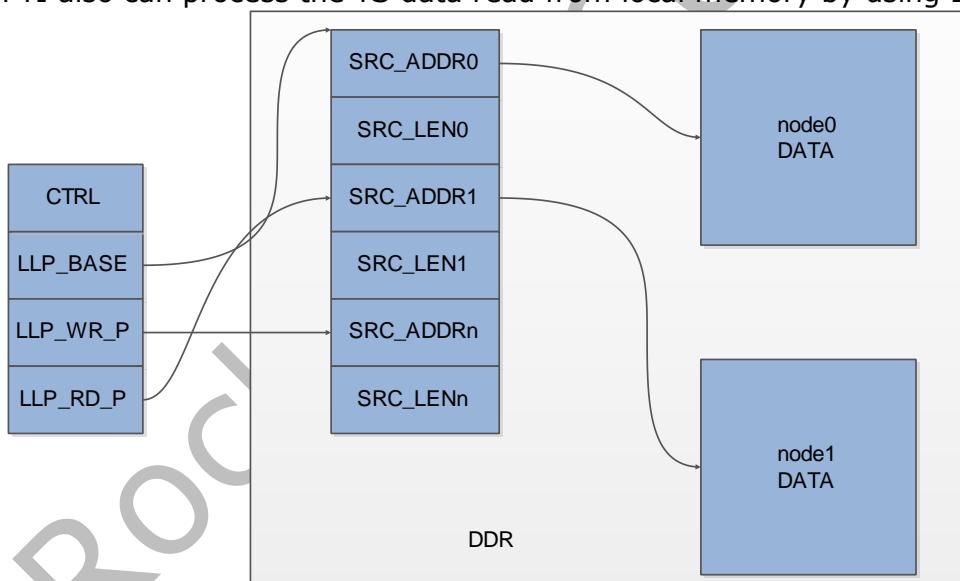
1.TS_IN Interface

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_LL_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_LL_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:

- A. 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 64th LLP node starting from the list base address,
- B. 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.
- C. 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.

13.5.3 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.

13.5.4 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.

13.5.5 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:

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.

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.

Demux data obtain

1. 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 1st , 4th to 18th 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.

Section Length = {First Word[11:8], First Word[23:16]};

Total Length = Section Length;

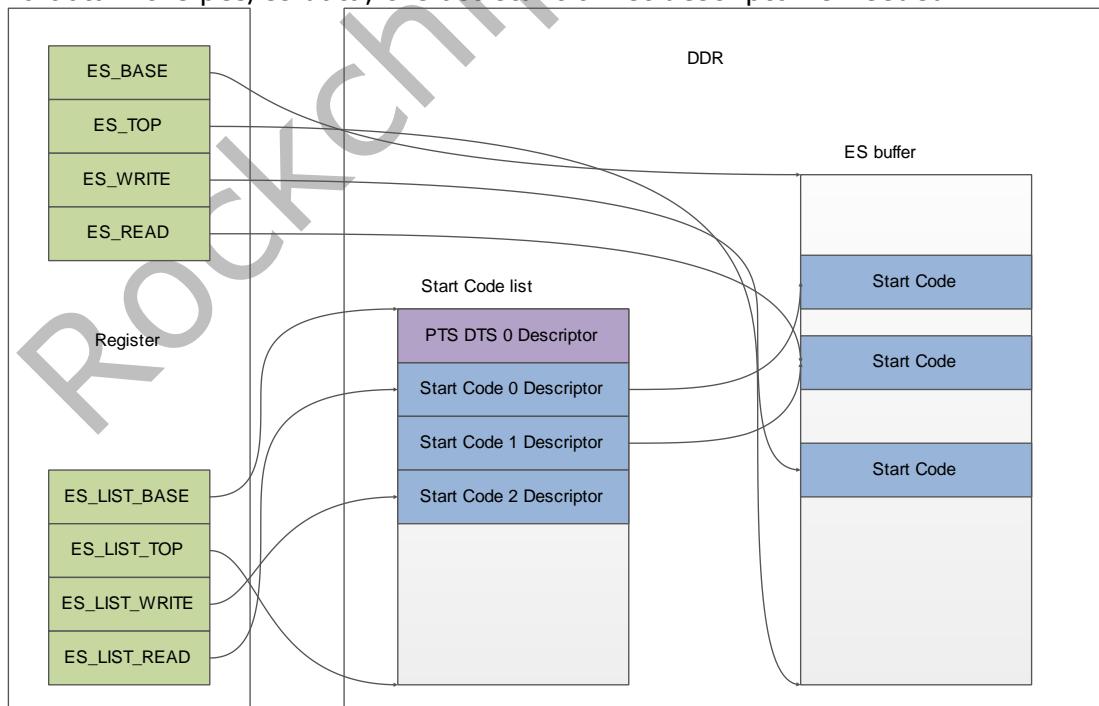
Then start to fetch section data according to the total length. Again don't forget to write the stopped address.

C. PES/ES filter

PES filter supports 16-byte filtering function, which can assign 4th, 7th to 21st 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:

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

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.

13.5.6 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.

13.5.7 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.

13.5.8 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.

Chapter 14 Temperature-Sensor ADC(TS-ADC)

14.1 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(high/low temperature) interrupt can be configurable
- In Automatic Mode, the temperature of system reset can be configurable
- Support to 1 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
- -40~125°C temperature range and 5°C temperature resolution
- 10-bit SARADC up to 50KS/s sampling rate

14.2 Block Diagram

TS-ADC controller comprises with:

- APB Interface
- TS-ADC control logic

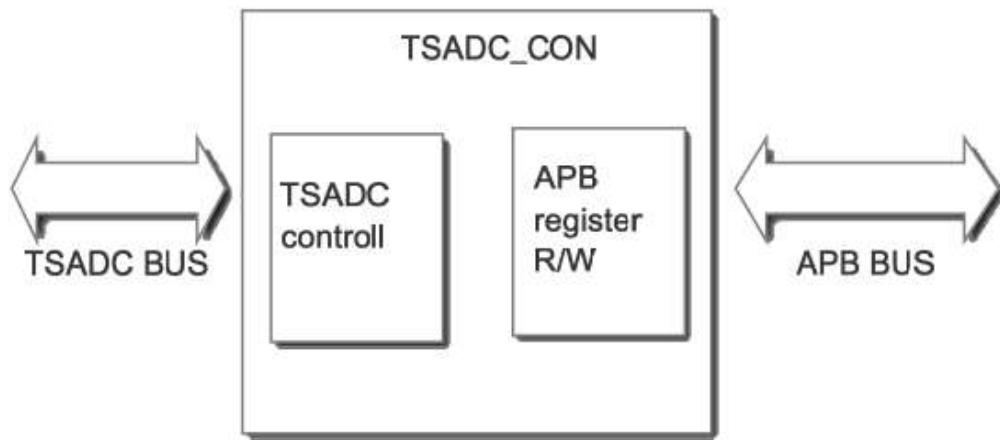


Fig. 10-33 TS-ADC Controller Block Diagram

14.3 Function Description

14.3.1 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.

14.3.2 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.

14.4 Register description

14.4.1 Registers Summary

| Name | Offset | Size | Reset Value | Description |
|----------------------------|--------|------|-------------|---|
| TSADC_USER_CON | 0x0000 | W | 0x00000208 | The control register of A/D Converter. |
| TSADC_AUTO_CON | 0x0004 | W | 0x00000000 | TSADC auto mode control register |
| TSADC_INT_EN | 0x0008 | W | 0x00000000 | |
| TSADC_INT_PD | 0x000c | W | 0x00000000 | |
| TSADC_DATA0 | 0x0020 | W | 0x00000000 | This register contains the data after A/D Conversion. |
| TSADC_COMP0_INT | 0x0030 | W | 0x00000000 | TSADC high temperature level for source 0 |
| TSADC_COMP0_SHUT | 0x0040 | W | 0x00000000 | TSADC high temperature level for source 0 |
| TSADC_HIGHT_INT_DEBOUNCE | 0x0060 | W | 0x00000003 | high temperature debounce |
| TSADC_HIGHT_TSHUT_DEBOUNCE | 0x0064 | W | 0x00000003 | high temperature debounce |
| TSADC_AUTO_PERIOD | 0x0068 | W | 0x00010000 | TSADC auto access period |
| TSADC_AUTO_PERIOD_H | 0x006c | W | 0x00010000 | TSADC auto access period when temperature is high |
| TSADC_COMP0_LOW_INT | 0x0080 | W | 0x00000000 | TSADC low temperature level for source 0 |

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

14.4.2 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. When TSADC_USER_CON[4] = 1'b1 take effect. |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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]). 000 : Input source 0 (SARADC_AIN[0]) 001 : Input source 1 (SARADC_AIN[1]) Others : Reserved |

TSADC_AUTO_CON

Address: Operational Base + offset (0x0004)

TSADC auto mode control register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 31:26 | RO | 0x0 | reserved |
| 25 | RW | 0x0 | last_tshut_2cru last_tshut_2cru for cru first/second reset 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. |
| 24 | RW | 0x0 | last_tshut_2gpio last_tshut_2gpio for hardware reset 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. |
| 15:13 | RO | 0x0 | reserved |
| 12 | RW | 0x0 | src0_lt_en 0: do not care low temperature of source 0 1: enable the low temperature monitor of source 0 |
| 11:9 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 8 | RW | 0x0 | tshut_polarity 0: low active 1: high active |
| 7:5 | RO | 0x0 | reserved |
| 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 |
| 3:2 | RO | 0x0 | reserved |
| 1 | RW | 0x0 | tsadc_q_sel whether select (1024 - tsadc_q) as output 1'b0:use tsadc_q as output(temperature-code is rising sequence) 1'b1:use(1024 - tsadc_q) as output (temperature-code is falling sequence) |
| 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)

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 31:17 | RO | 0x0 | reserved |
| 16 | RW | 0x0 | eoc_int_en eoc_Interrupt enable. eoc_interrupt enable in user defined mode 0: Disable; 1: Enable |
| 15:13 | RO | 0x0 | reserved |
| 12 | RW | 0x0 | lt_inten_src0 low temperature interrupt enable for src0 0: disable 1: enable |
| 11:9 | RO | 0x0 | reserved |
| 8 | RW | 0x0 | tshut_2cru_en_src0 0: TSHUT output to cru disabled. TSHUT output will always keep low . 1: TSHUT output works. |
| 7:5 | RO | 0x0 | reserved |
| 4 | RW | 0x0 | tshut_2gpio_en_src0 0: TSHUT output to gpio disabled. TSHUT output will always keep low . 1: TSHUT output works. |
| 3:1 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 0 | RW | 0x0 | ht_inten_src0 high temperature interrupt enable for src0 0: disable 1: enable |

TSADC_INT_PD

Address: Operational Base + offset (0x000c)

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:17 | RO | 0x0 | reserved |
| 16 | RW | 0x0 | eoc_int_pd Interrupt status. This bit will be set to 1 when end-of-conversion. Set 0 to clear the interrupt. |
| 15:13 | RO | 0x0 | reserved |
| 12 | RW | 0x0 | lt_irq_src0 When TSADC output is lower than COMP_INT_LOW, this bit will be valid, which means temperature is low, and the application should in charge of this. write 1 to it , this bit will be cleared. |
| 11:5 | RO | 0x0 | reserved |
| 4 | RW | 0x0 | tshut_o_src0 TSHUT output status When TSADC output is bigger than COMP_SHUT, this bit will be valid, which means temperature is VERY high, and the application should in charge of this. write 1 to it , this bit will be cleared. |
| 3:1 | RO | 0x0 | reserved |
| 0 | RW | 0x0 | ht_irq_src0 When TSADC output is bigger 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.

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:12 | RO | 0x0 | reserved |
| 11:0 | RO | 0x000 | adc_data A/D value of the channel 0 last conversion DOUT[9:0] |

TSADC_COMPO_INT

Address: Operational Base + offset (0x0030)

TSADC high temperature level for source 0

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:12 | RO | 0x0 | reserved |
| 11:0 | RW | 0x000 | tsadc_comp_src0 TSADC high temperature level. TSADC output is bigger than tsadc_comp, means the temperature is high. TSADC_INT will be valid. |

TSADC_COMP0_SHUT

Address: Operational Base + offset (0x0040)

TSADC high temperature level for source 0

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:12 | RO | 0x0 | reserved |
| 11:0 | RW | 0x000 | tsadc_comp_src0 TSADC high temperature level. TSADC output is bigger 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

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

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

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

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

TSADC_COMP0_LOW_INT

Address: Operational Base + offset (0x0080)

TSADC low temperature level for source 0

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:12 | RO | 0x0 | reserved |
| 11:0 | RW | 0x000 | tsadc_comp_src0 TSADC low temperature level. TSADC output is lower than tsadc_comp, means the temperature is low. TSADC_LOW_INT will be valid. |

14.5 Application Notes

14.5.1 Channel Select

The system has one Temperature Sensors, channel 0 is for CPU.

14.5.2 Single-sample conversion

To saving power, the TS-ADC used single-sample conversion. The timing as flowing picture:

- Bypass mode($grf_sco_con1[1] = 1'b1$)

When the ADC bypass mode is enable($tsadc_dig_bypass = 1'b1$), the ADC will cost 14 clock cycles to complete the conversion. The $tsadc_dout$ will keep the valid data output unchanged until the next clock cycles when the $tsadc_sample$ is valid.

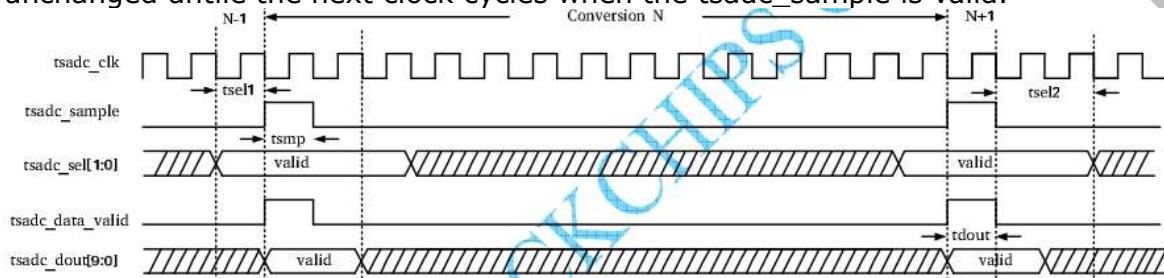


Fig. 10-34 tsadc timing diagram in bypass mode

- The Normal mode

- $tsadc_clk_sel = 1'b0$ ($grf_sco_con1[0] = 1'b0$)

The ADC will cost 15 clock cycles to complete the conversion. The $tsadc_dout$ will keep the valid data output unchanged until the next two clock cycles when the $tsadc_sample$ is valid.

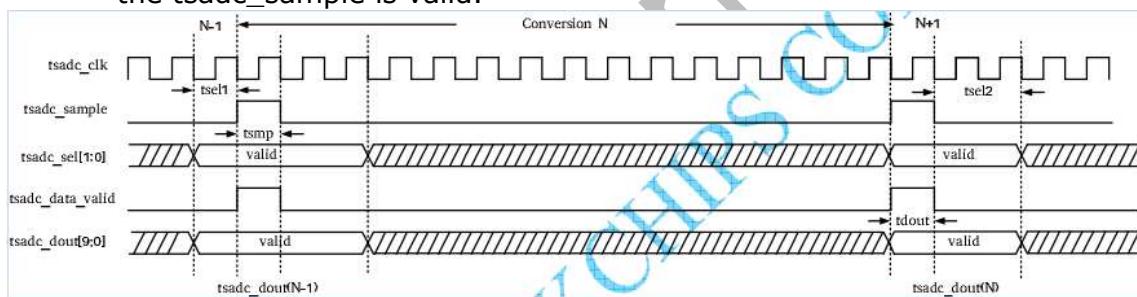


Fig. 10-35 tsadc timing diagram in normal mode with $tsadc_clk_sel = 1'b0$

- $tsadc_clk_sel = 1'b1$ ($grf_sco_con1[0] = 1'b1$)

The ADC will cost 16 clock cycles to complete the conversion. The $tsadc_dout$ will keep the valid data output unchanged until the next three clock cycles when the $tsadc_sample$ is valid.

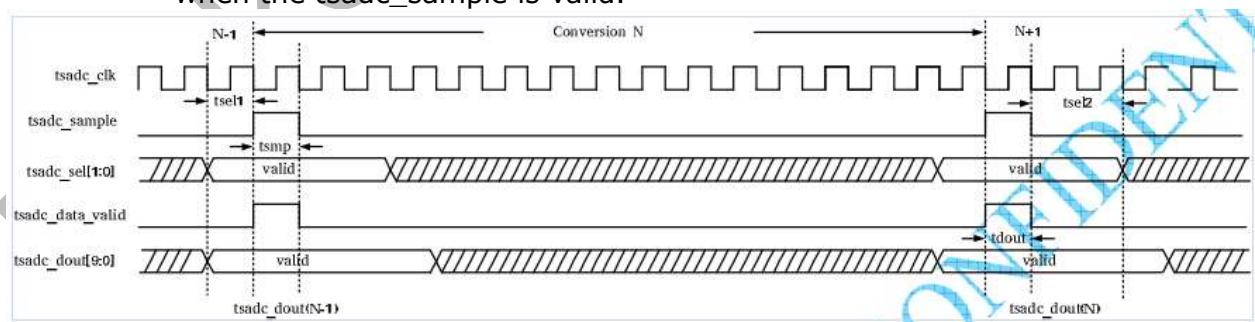


Fig. 10-36 tsadc timing diagram in normal mode with $tsadc_clk_sel = 1'b1$

14.5.3 Temperature-to-code mapping

Table 10-1 Temperature Code Mapping

| Temperature/°C | ADC Output Data | | |
|----------------|-----------------|-----|-----|
| | Min | Typ | Max |
| -40 | - | 436 | - |
| -35 | - | 431 | - |

| | | | |
|-----|---|-----|---|
| -30 | - | 426 | - |
| -25 | - | 421 | - |
| -20 | - | 416 | - |
| -15 | - | 411 | - |
| -10 | - | 406 | - |
| -5 | - | 401 | - |
| 0 | - | 395 | - |
| 5 | - | 390 | - |
| 10 | - | 385 | - |
| 15 | - | 380 | - |
| 20 | - | 375 | - |
| 25 | - | 370 | - |
| 30 | - | 364 | - |
| 35 | - | 359 | - |
| 40 | - | 354 | - |
| 45 | - | 349 | - |
| 50 | - | 343 | - |
| 55 | - | 338 | - |
| 60 | - | 333 | - |
| 65 | - | 328 | - |
| 70 | - | 322 | - |
| 75 | - | 317 | - |
| 80 | - | 312 | - |
| 85 | - | 307 | - |
| 90 | - | 301 | - |
| 95 | - | 296 | - |
| 100 | - | 291 | - |
| 105 | - | 286 | - |
| 110 | - | 280 | - |
| 115 | - | 275 | - |
| 120 | - | 270 | - |
| 125 | - | 264 | - |

Note:

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.

Code to Temperature mapping should be updated based on silicon results.

14.5.4 User-Define Mode

- 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.
- 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.
- Software can get the four channel temperature from TSADC_DATA_n (n=0,1,2,3).

14.5.5 Automatic Mode

You can use the automatic mode with the following step:

- Set TSADC_AUTO_PERIOD, configure the interleave between every two accessing of TSADC in normal operation.
- 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.
- Set TSADC_COMPn_INT(n=0,1), configure the high temperature level, if tsadc output is smaller than the value, means the temperature is high, tsadc_int will be asserted.
- Set TSADC_COMPn_SHUT(n=0,1), configure the super high temperature level, if tsadc output is smaller than the value, means the temperature is too high, TSHUT will be asserted.
- 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.
- 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.
- Set TSADC_AUTO_CON, enable the TSADC controller.

Chapter 15 GMAC Ethernet Interface

15.1 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.

15.1.1 Feature

- 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 de-assertion 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.

15.2 Block Diagram

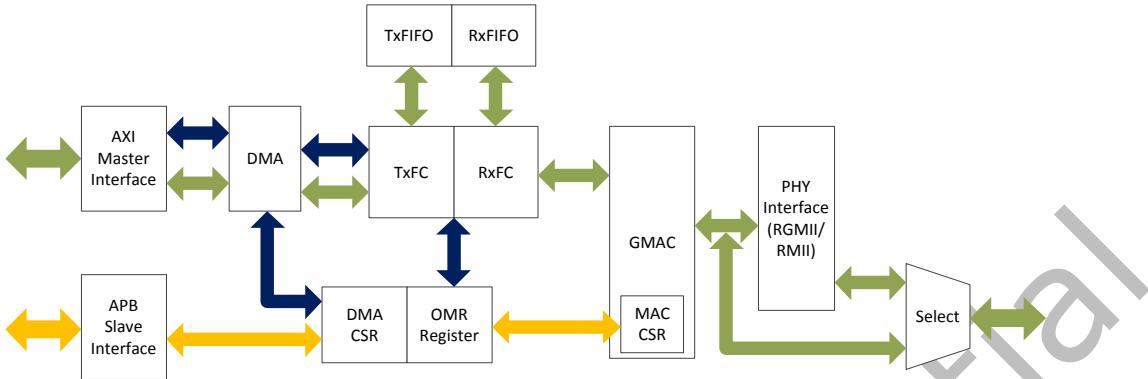


Fig. 10-37GMACArchitecture

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.

15.3 Function Description

15.3.1 Frame Structure

Data frames transmitted shall have the frame format shown in Fig. 25-2.

<inter-frame><preamble><sfd><data><efd>

Fig. 10-38MAC Block Diagram

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 octet's data.

15.3.2 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).

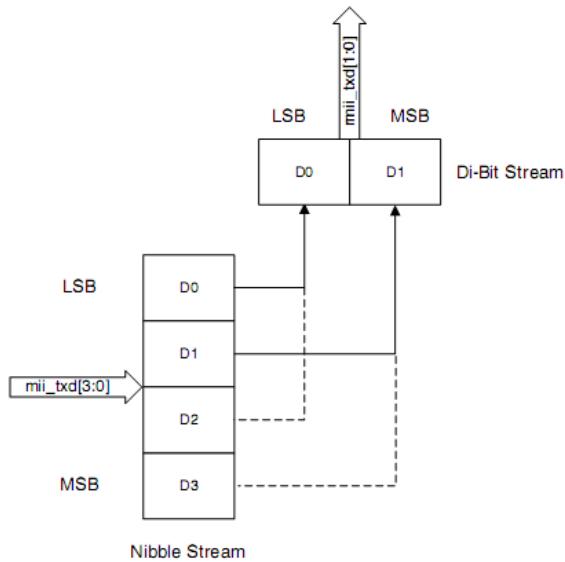


Fig. 10-39 RMII transmission bit ordering

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 group and yield the correct frame data.

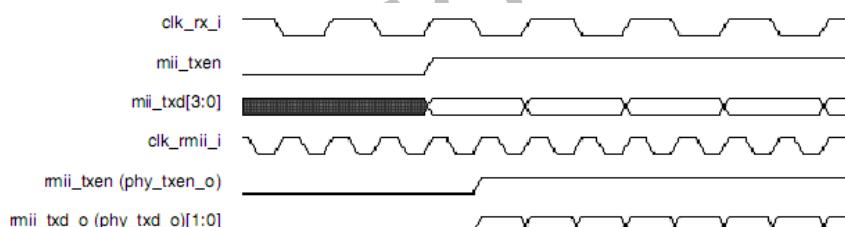


Fig. 10-40 Start of MII and RMII transmission in 100-Mbps mode

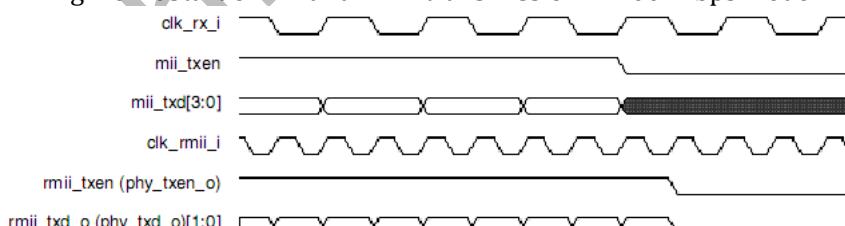


Fig. 10-41 End of MII and RMII Transmission in 100-Mbps Mode

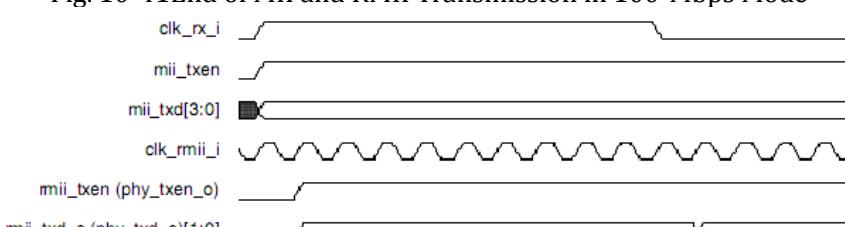


Fig. 10-42 Start of MII and RMII Transmission in 10-Mbps Mode

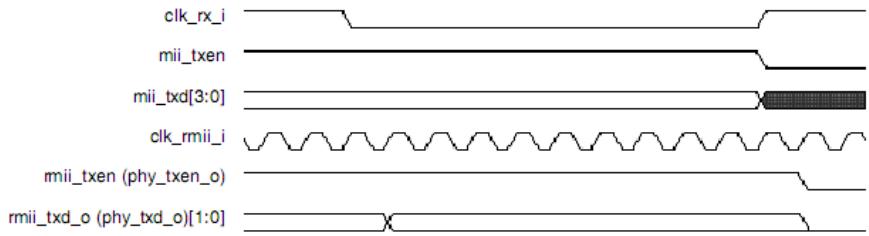


Fig. 10-43 End of MII and RMII Transmission in 10-Mbps Mode

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).

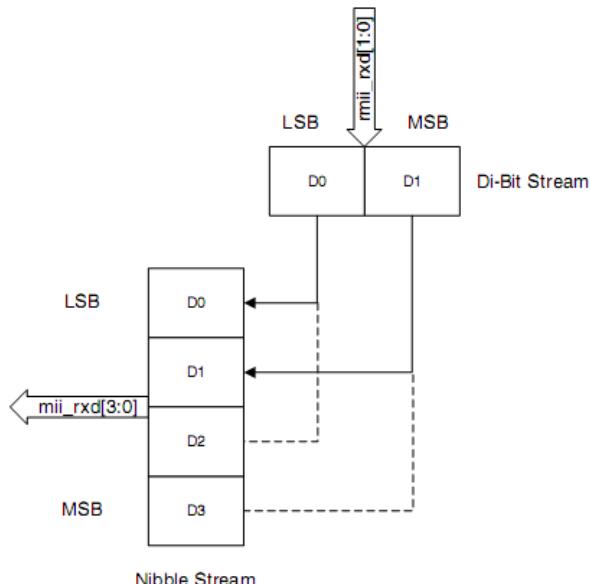


Fig. 10-44 RMII receive bit ordering

15.3.3 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 transmission 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.

15.3.4 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_mdi_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.

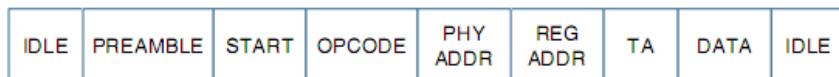


Fig. 10-45 MDIO frame structure

- 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.

15.3.5 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_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_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_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 FFFFFFFFFF
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.

15.3.6 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.

15.4 Register Description

15.4.1 Registers 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 |

| Name | Offset | Size | Reset Value | Description |
|------------------------|--------|------|-------------|--|
| 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 |
| 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_PART_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_INTR | 0x0104 | W | 0x00000000 | MMC Receive Interrupt Register |
| GMAC_MMC_TX_INTR | 0x0108 | W | 0x00000000 | MMC Transmit Interrupt Register |
| GMAC_MMC_RX_INT_MSK | 0x010c | W | 0x00000000 | MMC Receive Interrupt Mask Register |
| GMAC_MMC_TX_INT_MSK | 0x0110 | W | 0x00000000 | MMC Transmit Interrupt Mask Register |
| GMAC_MMC_TXOCTETCNT_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_TXOCTETCNT_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_RXOCTETCNT_GB | 0x0184 | W | 0x00000000 | MMC RX OCTET Good and Bad Counter |
| GMAC_MMC_RXOCTETCNT_G | 0x0188 | W | 0x00000000 | MMC RX OCTET Good Counter |
| GMAC_MMC_RXMCFRMCNT_G | 0x0190 | W | 0x00000000 | MMC RX Multicast Frame Good Counter |
| GMAC_MMC_RXCRCERR | 0x0194 | W | 0x00000000 | MMC RX Carrier |
| GMAC_MMC_RXLENERR | 0x01c8 | W | 0x00000000 | MMC RX Length Error |

| Name | Offset | Size | Reset Value | Description |
|---------------------------|--------|------|-------------|--|
| GMAC_MMC_RXFIFOVRF_LW | 0x01d4 | W | 0x00000000 | MMC RX FIFO Overflow |
| GMAC_MMC_IPC_INT_MS_K | 0x0200 | W | 0x00000000 | MMC Receive Checksum Offload Interrupt Mask Register |
| GMAC_MMC_IPC_INTR | 0x0208 | W | 0x00000000 | MMC Receive Checksum Offload Interrupt Register |
| GMAC_MMC_RXIPV4GFRM | 0x0210 | W | 0x00000000 | MMC RX IPV4 Good Frame |
| GMAC_MMC_RXIPV4HDER_RFRM | 0x0214 | W | 0x00000000 | MMC RX IPV4 Head Error Frame |
| GMAC_MMC_RXIPV6GFRM | 0x0224 | W | 0x00000000 | MMC RX IPV6 Good Frame |
| GMAC_MMC_RXIPV6HDER_RFRM | 0x0228 | W | 0x00000000 | MMC RX IPV6 Head Error Frame |
| GMAC_MMC_RXUDPERRFRM | 0x0234 | W | 0x00000000 | MMC RX UDP Error Frame |
| GMAC_MMC_RXTCPERRFRM | 0x023c | W | 0x00000000 | MMC RX TCP Error Frame |
| GMAC_MMC_RXICMPERRFRM | 0x0244 | W | 0x00000000 | MMC RX ICMP Error Frame |
| GMAC_MMC_RXIPV4HDER_ROCT | 0x0254 | W | 0x00000000 | MMC RX OCTET IPV4 Head Error |
| GMAC_MMC_RXIPV6HDER_ROCT | 0x0268 | W | 0x00000000 | MMC RX OCTET IPV6 Head Error |
| GMAC_MMC_RXUDPERROCT | 0x0274 | W | 0x00000000 | MMC RX OCTET UDP Error |
| GMAC_MMC_RXTCPPERROCT | 0x027c | W | 0x00000000 | MMC RX OCTET TCP Error |
| GMAC_MMC_RXICMPERR_OCT | 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_A_DDR | 0x100c | W | 0x00000000 | Receive Descriptor List Address Register |
| GMAC_TX_DESC_LIST_ADDRESS | 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 |

| Name | Offset | Size | Reset Value | Description |
|------------------------------|--------|------|-------------|---|
| GMAC_CUR_HOST_TX_DESCRIPTOR | 0x1048 | W | 0x00000000 | Current Host Transmit Descriptor Register |
| GMAC_CUR_HOST_RX_DESCRIPTOR | 0x104c | W | 0x00000000 | Current Host Receive Descriptor Register |
| GMAC_CUR_HOST_TX_BUFFER_ADDR | 0x1050 | W | 0x00000000 | Current Host Transmit Buffer Address Register |
| GMAC_CUR_HOST_RX_BUFFER_ADDR | 0x1054 | W | 0x00000000 | Current Host Receive Buffer Address Register |

Notes: **S**-ize: **B**- Byte (8 bits) access, **H****W**- Half WORD (16 bits) access, **W**-WORD (32 bits) access

15.4.2 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 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. |
| 23 | RW | 0x0 | 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. |
| 22 | RW | 0x0 | 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. |
| 21 | RW | 0x0 | BE Frame Burst Enable When this bit is set, the GMAC allows frame bursting during transmission in GMII Half-Duplex mode. |
| 20 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 19:17 | RW | 0x0 | <p>IFG Inter-Frame Gap These bits control the minimum IFG between frames during transmission.</p> <p>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> |
| 15 | RW | 0x0 | <p>PS Port Select Selects between GMII and MII: 1'b0: GMII (1000 Mbps) 1'b1: MII (10/100 Mbps)</p> |
| 14 | RW | 0x0 | <p>FES Speed Indicates the speed in Fast Ethernet (MII) mode: 1'b0: 10 Mbps 1'b1: 100 Mbps</p> |
| 13 | RW | 0x0 | <p>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.</p> |
| 12 | RW | 0x0 | <p>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.</p> |
| 11 | RW | 0x0 | <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 10 | RW | 0x0 | <p>IPC Checksum Offload</p> <p>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).</p> <p>When this bit is reset, this function is disabled.</p> <p>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</p> <p>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.</p> <p>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</p> <p>Indicates whether the link is up or down during the transmission of configuration in RGMII interface:</p> <p>1'b0: Link Down 1'b1: Link Up</p> |
| 7 | RW | 0x0 | <p>ACS Automatic Pad/CRC Stripping</p> <p>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> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 6:5 | RW | 0x0 | <p>BL Back-Off Limit</p> <p>The Back-Off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps and 512 bit times for 10/100 Mbps) the 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 < 2^k</p> |
| 4 | RW | 0x0 | <p>DC Deferral Check</p> <p>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 transmission 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</p> <p>When this bit is set, the transmission 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</p> <p>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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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.</p> <p>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> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. 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. 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. When reset, normal filtering of frames is performed.</p> |
| 2 | RW | 0x0 | <p>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.</p> |
| 1 | RW | 0x0 | <p>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.</p> |

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

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

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

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

| Bit | Attr | Reset Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|------------|--------------|---------------|--|-----------|-----------|-----------|------|------------|--------------|------|-------------|--------------|------|-----------|--------------|------|-----------|--------------|------|-------------|---------------|------|-------------|---------------|------------|----------|--|-----------|-----------|------|-------------|------|-------------|------|-------------|------|--------------|------|--------------|------|--------------|------|--------------|------|--------------|
| 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>pclk_gmac</th> <th>MDC Clock</th> </tr> </thead> <tbody> <tr> <td>0000</td> <td>60-100 MHz</td> <td>pclk_gmac/42</td> </tr> <tr> <td>0001</td> <td>100-150 MHz</td> <td>pclk_gmac/62</td> </tr> <tr> <td>0010</td> <td>20-35 MHz</td> <td>pclk_gmac/16</td> </tr> <tr> <td>0011</td> <td>35-60 MHz</td> <td>pclk_gmac/26</td> </tr> <tr> <td>0100</td> <td>150-250 MHz</td> <td>pclk_gmac/102</td> </tr> <tr> <td>0101</td> <td>250-300 MHz</td> <td>pclk_gmac/124</td> </tr> <tr> <td>0110, 0111</td> <td>Reserved</td> <td></td> </tr> </tbody> </table> <p>When bit 5 is set, you can achieve MDC clock of frequency higher than the IEEE802.3 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 | 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 | | 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 | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | RW | 0x0 | <p>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.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 0 | W1C | 0x0 | <p>GB GMII Busy</p> <p>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.</p> |

GMAC_GMII_DATA

Address: Operational Base + offset (0x0014)

GMII Data Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 31:16 | RO | 0x0 | reserved |
| 15:0 | RW | 0x0000 | <p>GD GMII Data</p> <p>This contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.</p> |

GMAC_FLOW_CTRL

Address: Operational Base + offset (0x0018)

Flow Control Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 31:16 | RW | 0x0000 | <p>PT Pause Time</p> <p>This field holds the value to be used in the Pause Time field in the transmit control frame. If the Pause Time bits is configured to be double-synchronized to the (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</p> <p>When set, this bit disables the automatic generation of Zero-Quanta Pause Control frames on the de-assertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i/mti_flowctrl_i).</p> <p>When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled.</p> |
| 6 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description | | | | | | | | | | |
|-----------|---------------------------------|-------------|---|-----------|-----------|----|-------------------------------|----|--------------------------------|----|---------------------------------|----|---------------------------------|
| 5:4 | RW | 0x0 | <p>PLT Pause Low Threshold</p> <p>This field configures the threshold of the PAUSE timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic 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 | | | | | | | | | | | | |
| 3 | RW | 0x0 | <p>UP Unicast Pause Frame Detect</p> <p>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.</p> | | | | | | | | | | |
| 2 | RW | 0x0 | <p>RFE Receive Flow Control Enable</p> <p>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.</p> | | | | | | | | | | |
| 1 | RW | 0x0 | <p>TFE Transmit Flow Control Enable</p> <p>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.</p> <p>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.</p> | | | | | | | | | | |

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

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 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> |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:26 | RO | 0x0 | reserved |
| 25 | RW | 0x0 | 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. |
| 24 | RW | 0x0 | TFIFO2 When high, it indicates that the MTL TxFIFO is not empty and has some data left for transmission. |
| 23 | RO | 0x0 | reserved |
| 22 | RW | 0x0 | TFIFO1 When high, it indicates that the MTL TxFIFO Write Controller is active and transferring data to the TxFIFO. |
| 21:20 | RW | 0x0 | 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 |
| 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 |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 6:5 | RW | 0x0 | <p>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</p> |
| 4 | RW | 0x0 | <p>RFIFOWR When high, it indicates that the MTL RxFIFO Write Controller is active and transferring a received frame to the FIFO.</p> |
| 3 | RO | 0x0 | reserved |
| 2:1 | RW | 0x0 | <p>ACT When high, it indicates the active state of the small FIFO Read and Write controllers respectively of the MAC receive Frame Controller module</p> |
| 0 | RW | 0x0 | <p>RDB When high, it indicates that the MAC GMII/MII receive protocol engine is actively receiving data and not in IDLE state.</p> |

GMAC_PMT_CTRL_STA

Address: Operational Base + offset (0x002c)

PMT Control and Status Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 31 | W1C | 0x0 | <p>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.</p> |
| 30:10 | RO | 0x0 | reserved |
| 9 | RW | 0x0 | <p>GU Global Unicast When set, enables any unicast packet filtered by the GMAC (DAF) address recognition to be a wake-up frame.</p> |
| 8:7 | RO | 0x0 | reserved |
| 6 | RC | 0x0 | <p>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.</p> |
| 5 | RC | 0x0 | <p>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.</p> |
| 4:3 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 2 | RW | 0x0 | WFE Wake-Up Frame Enable When set, enables generation of a power management event due to wake-up frame reception. |
| 1 | RW | 0x0 | MPE Magic Packet Enable When set, enables generation of a power management event due to Magic Packet reception. |
| 0 | R/W SC | 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 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. |

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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 | R/W SC | 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:6 | RO | 0x0 | reserved |
| 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 | R/W SC | 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:4 | RO | 0x0 | reserved |
| 3 | RO | 0x0 | LST Link Status Indicates whether the link is up (1'b1) or down (1'b0) |
| 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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 - 2K Bytes) 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 - 2K Bytes) and all frame-counters gets preset to 0xFFFF_FFF0 (full - 16) |
| 4 | R/W SC | 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. |

| Bit | Attr | Reset Value | Description |
|-----|--------|-------------|---|
| 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 | R/W SC | 0x0 | CR Counters Reset When set, all counters will be reset. This bit will be cleared automatically after 1 clock cycle |

GMAC_MMCRX_INTR

Address: Operational Base + offset (0x0104)

MMC Receive Interrupt Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 31:22 | RO | 0x0 | reserved |
| 21 | RW | 0x0 | INT21 The bit is set 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 | RC | 0x0 | INT18 The bit is set 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 The bit is set when the rx crc error counter reaches half the maximum value, and also when it reaches the maximum value. |
| 4 | RC | 0x0 | INT4 The bit is set 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 | RC | 0x0 | INT2 The bit is set when the rx octet count_g counter reaches half the maximum value, and also when it reaches the maximum value. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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_MMCTXINTR

Address: Operational Base + offset (0x0108)

MMC Transmit Interrupt Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. |
| 0 | RC | 0x0 | INT0 The bit is set when the rxoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value. |

GMAC_MMCRXINTMSK

Address: Operational Base + offset (0x010c)

MMC Receive Interrupt Mask Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:22 | RO | 0x0 | reserved |
| 21 | RW | 0x0 | INT21 Setting this bit masks the interrupt when the rxfifooverflow counter reaches half the maximum value, and also when it reaches the maximum value. |
| 20:19 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 18 | RW | 0x0 | INT18 Setting this bit masks the interrupt 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 Setting this bit masks the interrupt when the rxcrcerror 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 rxmulticastframes_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 rxoctetcount_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 rxoctetcount_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 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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RW | 0x00000000 | txframecount_gb Number of good and bad frames transmitted, exclusive of retried frames. |

GMAC_MMC_TXUNDLFLWERR

Address: Operational Base + offset (0x0148)

MMC TX Underflow Error

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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 31:30 | RO | 0x0 | reserved |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. |
| 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

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

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

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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RW | 0x00000000 | rxicmp_err_frms Number of good IP datagrams whose ICMP payload has a checksum error. |

GMAC_MMC_RXIPV4HDERRROCT

Address: Operational Base + offset (0x0254)

MMC RX OCTET IPV4 Head Error

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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_RXIPV6HDERRROCT

Address: Operational Base + offset (0x0268)

MMC RX OCTET IPV6 Head Error

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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_RXUDPPERROCT

Address: Operational Base + offset (0x0274)

MMC RX OCTET UDP Error

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31:0 | RW | 0x00000000 | rxtcp_err_octets Number of bytes received in a TCP segment with checksum errors. |

GMAC_MMICMPERRCOUNT

Address: Operational Base + offset (0x0284)

MMC RX OCTET ICMP Error

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 16 | RW | 0x0 | <p>FB Fixed Burst</p> <p>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.</p> |
| 15:14 | RO | 0x0 | reserved |
| 13:8 | RW | 0x01 | <p>PBL Programmable Burst Length</p> <p>These bits indicate the maximum number of beats to be transferred in one DMA transaction. This will be the maximum value that is used in a single block Read/Write.</p> <p>The DMA will always attempt to burst as specified in PBL each time it starts a Burst transfer on the host bus. PBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value will result in undefined behavior. When USP is set high, this PBL value is applicable for TxDMA transactions only.</p> <p>The PBL values have the following limitations.</p> <p>The maximum number of beats (PBL) possible is limited by the size of the Tx FIFO and Rx FIFO in the MTL layer and the data bus width on the DMA. The FIFO has a constraint that the maximum beat supported is half the depth of the FIFO, 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</p> <p>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 |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 0 | R/W SC | 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

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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 25:23 | RO | 0x0 | <p>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.</p> <p>Bit 23: 1'b1 Error during data transfer by TxDMA 1'b0 Error during data transfer by RxDMA</p> <p>Bit 24: 1'b1 Error during read transfer 1'b0 Error during write transfer</p> <p>Bit 25: 1'b1 Error during descriptor access 1'b0 Error during data buffer access</p> |
| 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. 3'b001: Running; Fetching Transmit Transfer Descriptor. 3'b010: Running; Waiting for status. 3'b011: Running; Reading Data from host memory buffer and queuing it to transmit buffer (Tx FIFO). 3'b100: TIME_STAMP write state. 3'b101: Reserved for future use. 3'b110: Suspended; Transmit Descriptor Unavailable or Transmit Buffer Underflow. 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. 3'b001: Running: Fetching Receive Transfer Descriptor. 3'b010: Reserved for future use. 3'b011: Running: Waiting for receive packet. 3'b100: Suspended: Receive Descriptor Unavailable. 3'b101: Running: Closing Receive Descriptor. 3'b110: TIME_STAMP write state. 3'b111: Running: Transferring the receive packet data from receive buffer to host memory.</p> |

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 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> <p>Register GMAC_STATUS[0]: Transmit Interrupt Register GMAC_STATUS[2]: Transmit Buffer Unavailable Register GMAC_STATUS[6]: Receive Interrupt Register GMAC_STATUS[14]: Early Receive Interrupt 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> |
| 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> <p>Register GMAC_STATUS[1]: Transmit Process Stopped Register GMAC_STATUS[3]: Transmit Jabber Timeout Register GMAC_STATUS[4]: Receive FIFO Overflow Register GMAC_STATUS[5]: Transmit Underflow Register GMAC_STATUS[7]: Receive Buffer Unavailable Register GMAC_STATUS[8]: Receive Process Stopped Register GMAC_STATUS[9]: Receive Watchdog Timeout Register GMAC_STATUS[10]: Early Transmit Interrupt Register GMAC_STATUS[13]: Fatal Bus Error Only unmasked bits affect the Abnormal Interrupt Summary bit. 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 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 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 This bit indicates that the frame to be transmitted was fully transferred to the MTL Transmit FIFO.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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]. |
| 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. |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 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 TDTS1[31] is set in the First Descriptor. |

GMAC_OP_MODE

Address: Operational Base + offset (0x1018)

Operation Mode Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 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. |
| 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 |

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 21 | RW | 0x0 | <p>TSF Transmit Store and Forward</p> <p>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.</p> |
| 20 | W1C | 0x0 | <p>FTF Flush Transmit FIFO</p> <p>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.</p> <p>Note: The flush operation completes only after emptying the TxFIFO 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.</p> |
| 19:17 | RO | 0x0 | reserved |
| 16:14 | RW | 0x0 | <p>TTC Transmit Threshold Control</p> <p>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> <p>3'b000: 64 3'b001: 128 3'b010: 192 3'b011: 256 3'b100: 40 3'b101: 32 3'b110: 24 3'b111: 16</p> |

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 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 de-asserted 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</p> <p>Note that the de-assertion is effective only after flow control is asserted.</p> |
| 10:9 | RW | 0x0 | <p>RFA Threshold for activating flow control (in both HD and FD) 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 Enable HW flow control 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> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 7 | RW | 0x0 | <p>FEF Forward Error Frames 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. When FEF is set, all frames except runt error frames are forwarded to the DMA. But when Rx FIFO overflows when a partial frame is written, then such frames are dropped even when FEF is set.</p> |
| 6 | RW | 0x0 | <p>FUF Forward Undersized Good Frames 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 |
| 4:3 | RW | 0x0 | <p>RTC Receive Threshold Control These two bits control the threshold level of the MTL Receive FIFO. Transfer (request) to DMA starts when the frame size within the MTL Receive FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are transferred automatically. Note that value of 11 is not applicable if the configured Receive FIFO size is 128 bytes. These bits are valid only when the RSF bit is zero, and are ignored when the RSF bit is set to 1. 2'b00: 64 2'b01: 32 2'b10: 96 2'b11: 128</p> |
| 2 | RW | 0x0 | <p>OSF Operate on Second Frame 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> |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 1 | RW | 0x0 | <p>SR Start/Stop Receive</p> <p>When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes incoming frames. Descriptor acquisition is attempted from the current position in the list, which is the address set by 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.</p> <p>When this bit is cleared, RxDMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state.</p> |
| 0 | RO | 0x0 | reserved |

GMAC_INT_ENA

Address: Operational Base + offset (0x101c)

Interrupt Enable Register

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|--|
| 31:17 | RO | 0x0 | reserved |
| 16 | RW | 0x0 | <p>NIE Normal Interrupt Summary Enable</p> <p>When this bit is set, a normal interrupt is enabled. When this bit is reset, a normal interrupt is disabled. This bit enables the following bits:</p> <ul style="list-style-type: none"> Register GMAC_STATUS[0]: Transmit Interrupt Register GMAC_STATUS[2]: Transmit Buffer Unavailable Register GMAC_STATUS[6]: Receive Interrupt Register GMAC_STATUS[14]: Early Receive Interrupt |

| Bit | Attr | Reset Value | Description |
|-------|------|-------------|---|
| 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> <p>Register GMAC_STATUS[1]: Transmit Process Stopped Register GMAC_STATUS[3]: Transmit Jabber Timeout Register GMAC_STATUS[4]: Receive Overflow Register GMAC_STATUS[5]: Transmit Underflow Register GMAC_STATUS[7]: Receive Buffer Unavailable Register GMAC_STATUS[8]: Receive Process Stopped Register GMAC_STATUS[9]: Receive Watchdog Timeout Register GMAC_STATUS[10]: Early Transmit Interrupt Register GMAC_STATUS[13]: Fatal Bus Error</p> |
| 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> |
| 13 | RW | 0x0 | <p>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.</p> |
| 12:11 | RO | 0x0 | reserved |
| 10 | RW | 0x0 | <p>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.</p> |
| 9 | RW | 0x0 | <p>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.</p> |
| 8 | RW | 0x0 | <p>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.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 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. |
| 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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:29 | RO | 0x0 | reserved |
| 28 | RC | 0x0 | FIFO_overflow_bit Overflow bit for FIFO Overflow Counter |
| 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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 31 | RW | 0x0 | 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 30 | RW | 0x0 | 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. |
| 29:22 | RO | 0x0 | reserved |
| 21:20 | RW | 0x1 | 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 |
| 19:18 | RO | 0x0 | reserved |
| 17:16 | RW | 0x1 | 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 |
| 15:13 | RO | 0x0 | reserved |
| 12 | RO | 0x0 | AXI_AAL Address-Aligned Beats This bit is read-only bit and reflects the AAL bit (register GMAC_BUS_MODE[25]). When this bit set to 1, it performs address-aligned burst transfers on both read and write channels. |
| 11:4 | RO | 0x0 | reserved |
| 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. |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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, 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

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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 31:0 | RO | 0x00000000 | HRBAP Host Receive Buffer Address Pointer Cleared on Reset. Pointer updated by DMA during operation. |

15.5 Interface Description

Table 10-7 RMII Interface Description

| Module pin | Direction | Pad name | IOMUX setting |
|----------------------|------------------|---|-------------------------------|
| RMII interface | | | |
| mac_clk | I/O | IO_MACclk_CIFclkout_DVPTSGpio2b3 | GRF_GPIO2B_IOMUX[7:6]=2'b01 |
| mac_txen | O | IO_MACtxen_CIFdata0_ISPflashtrigin_D VPTSGpio2b4 | GRF_GPIO2B_IOMUX[9:8]=2'b01 |
| mac_txd1 | O | IO_MACtxd1_CIFdata3_DVPTSGpio2a1 | GRF_GPIO2A_IOMUX[3:2]=2'b01 |
| mac_txd0 | O | IO_MACtxd0_CIFdata2_DVPTSGpio2a0 | GRF_GPIO2A_IOMUX[1:0]=2'b01 |
| mac_rxdv | I | IO_MACrxdv_CIFhref_DVPTSGpio2b1 | GRF_GPIO2B_IOMUX[3:2]=2'b01 |
| mac_rxer | I | IO_MACrxer_CIFclkin_DVPTSGpio2b2 | GRF_GPIO2B_IOMUX[5:4]=2'b01 |
| mac_rxd1 | I | IO_MACrxd1_CIFdata5_DVPTSGpio2a3 | GRF_GPIO2A_IOMUX[7:6]=2'b01 |
| mac_rxd0 | I | IO_MACrxd0_CIFdata4_DVPTSGpio2a2 | GRF_GPIO2A_IOMUX[5:4]=2'b01 |
| Management interface | | | |
| mac_mdio | I/O | IO_MACmdio_CIFdata1_ISPshuttertrig_D VPTSGpio2b5 | GRF_GPIO2B_IOMUX[11:10]=2'b01 |
| mac_mdc | O | IO_MACmdc_CIFvsync_DVPTSGpio2b0 | GRF_GPIO2B_IOMUX[1:0]=2'b01 |

Table 10-8 RGMII Interface Description

| Module pin | Direction | Pad name | IOMUX setting |
|----------------------|------------------|---|-------------------------------|
| RGMII/RMII interface | | | |
| mac_clk | I/O | IO_MACclk_CIFclkout_DVPTSGpio2b3 | GRF_GPIO2B_IOMUX[7:6]=2'b01 |
| mac_txclk | O | IO_MACtxclkout_CAMI2C3sda_DVPTSGpio2c1 | GRF_GPIO2C_IOMUX[3:2]=2'b01 |
| mac_txen | O | IO_MACtxen_CIFdata0_ISPflashtrigin_D VPTSGpio2b4 | GRF_GPIO2B_IOMUX[9:8]=2'b01 |
| mac_txd3 | O | IO_MACtxd3_CIFdata7_SPI1csn0t0_DVPTSGpio2a5 | GRF_GPIO2A_IOMUX[11:10]=2'b01 |
| mac_txd2 | O | IO_MACtxd2_CIFdata6_SPI1clk0_DVPTSGpio2a4 | GRF_GPIO2A_IOMUX[9:8]=2'b01 |
| mac_txd1 | O | IO_MACtxd1_CIFdata3_DVPTSGpio2a1 | GRF_GPIO2A_IOMUX[3:2]=2'b01 |
| mac_txd0 | O | IO_MACtxd0_CIFdata2_DVPTSGpio2a0 | GRF_GPIO2A_IOMUX[1:0]=2'b01 |
| mac_rxclk | I | IO_MACrxclkin_CIFdata10_ISPshutteren _DVPTSGpio2b6 | GRF_GPIO2B_IOMUX[13:12]=2'b01 |
| mac_rxdv | I | IO_MACrxdv_CIFhref_DVPTSGpio2b1 | GRF_GPIO2B_IOMUX[3:2]=2'b01 |
| mac_rxd3 | I | IO_MACrxd3_CIFdata9_SPI1rxdt0_DVPTSGpio2a7 | GRF_GPIO2A_IOMUX[15:14]=2'b01 |
| mac_rxd2 | I | IO_MACrxd2_CIFdata8_SPI1txd_DVPTSGpio2a6 | GRF_GPIO2A_IOMUX[13:12]=2'b01 |
| mac_rxd1 | I | IO_MACrxd1_CIFdata5_DVPTSGpio2a3 | GRF_GPIO2A_IOMUX[7:6]=2'b01 |
| mac_rxd0 | I | IO_MACrxd0_CIFdata4_DVPTSGpio2a2 | GRF_GPIO2A_IOMUX[5:4]=2'b01 |
| mac_crs | I | IO_MACcrs_CIFdata11_PWM2t0_DVPTSG | GRF_GPIO2B_IOMUX[15:14]=2'b01 |

| | | | |
|----------------------|-----|---|-------------------------------|
| | | pio2b7 | 01 |
| mac_col | I | IO_MACcol_CAMI2C3scl_DVPTSGpio2c0 | GRF_GPIO2C_IOMUX[1:0]=2'b01 |
| Management interface | | | |
| mac_mdio | I/O | IO_MACmdio_CIFdata1_ISPshuttertrig_D VPTSGpio2b5 | GRF_GPIO2B_IOMUX[11:10]=2'b01 |
| mac_mdc | O | IO_MACmdc_CIFvsync_DVPTSGpio2b0 | GRF_GPIO2B_IOMUX[1:0]=2'b01 |

Notes: I=input, O=output, I/O=input/output, bidirectional

15.6 Application Notes

15.6.1 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.

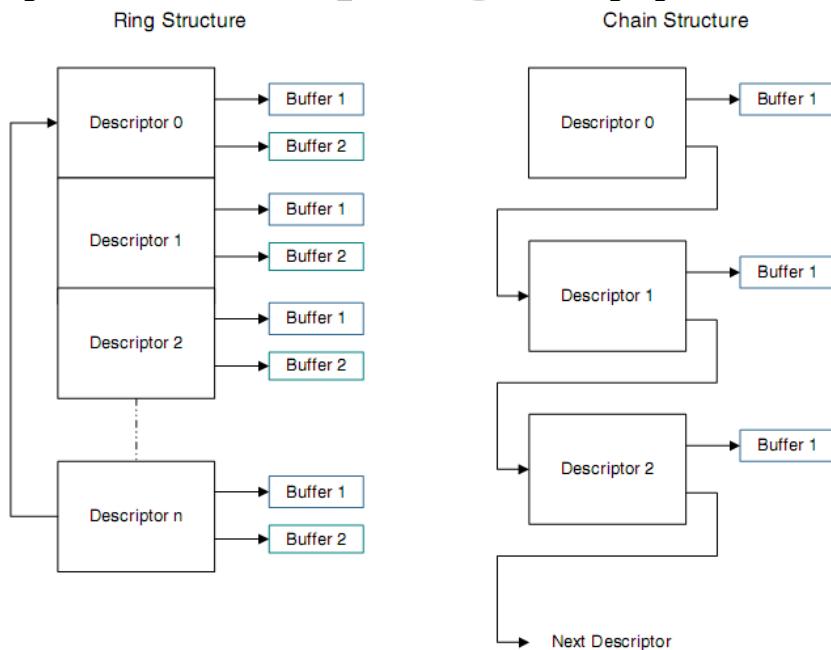


Fig. 10-46Descriptor Ring and Chain Structure

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).

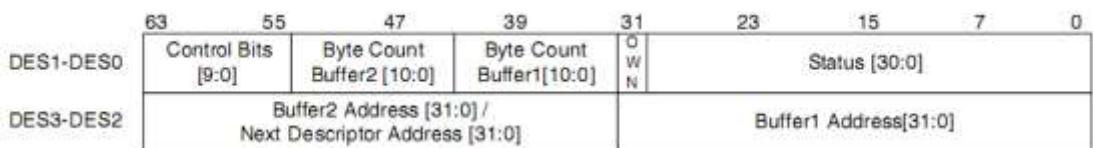


Fig. 10-47 Rx/Tx Descriptors definition

15.6.2 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 10-9 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"> • RDES0[0]: Payload Checksum Error • RDES0[1]: CRC Error • RDES0[3]: Receive Error • RDES0[4]: Watchdog Timeout • RDES0[6]: Late Collision • RDES0[7]: IPC Checksum • RDES0[11]: Overflow Error • RDES0[14]: Descriptor Error 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 GMAC Core. |
| 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_rxdv_i is asserted during frame reception. This error also includes carrier extension error in GMII and Half-duplex mode. Error can be of less/no extension, or error ($rxd \neq 0f$) 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 10-10Receive 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 10-11Receive 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 10-12Receive Descriptor 3

| Bit | Description |
|------|---|
| 31:0 | 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. |

| | |
|--|---|
| | 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. |
|--|---|

15.6.3 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 10-13 Transmit Descriptor 0

| Bit | Description |
|-------|---|
| 31 | 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. |
| 30:17 | Reserved. |
| 16 | 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. |
| 15 | ES: Error Summary Indicates the logical OR of the following bits: <ul style="list-style-type: none"> • TDES0[14]: Jabber Timeout • TDES0[13]: Frame Flush • TDES0[11]: Loss of Carrier • TDES0[10]: No Carrier • TDES0[9]: Late Collision • TDES0[8]: Excessive Collision • TDES0[2]: Excessive Deferral • TDES0[1]: Underflow Error |
| 14 | JT: Jabber Timeout When set, this bit indicates the GMAC transmitter has experienced a jabber time-out. |
| 13 | FF: Frame Flushed 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 10-14 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"> • 2'b00: Do nothing. Checksum Engine is bypassed • 2'b01: Insert IPv4 header checksum. Use this value to insert IPv4 header checksum when the frame encapsulates an IPv4 datagram. • 2'b10: Insert TCP/UDP/ICMP checksum. The checksum is calculated over the TCP, |

| | |
|-------|--|
| | <p>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.</p> <ul style="list-style-type: none"> • 2'b11: Insert a TCP/UDP/ICMP checksum that is fully calculated in this engine. In other words, the TCP, UDP, or ICMP pseudo-header is included in the checksum calculation, and the input frame's corresponding Checksum field has an all-zero value. An IPv4 Header checksum is also inserted if the encapsulated datagram conforms to IPv4. <p>The Checksum engine detects whether the TCP, UDP, or ICMP segment is encapsulated in IPv4 or IPv6 and processes its data accordingly.</p> |
| 26 | <p>DC: Disable CRC</p> <p>When set, the 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]).</p> |
| 25 | <p>TER: Transmit End of Ring</p> <p>When set, this bit indicates that the descriptor list reached its final descriptor. The returns to the base address of the list, creating a descriptor ring.</p> |
| 24 | <p>TCH: Second Address Chained</p> <p>When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES1[24] is set, TBS2 (TDES1[21-11]) are "don't care" values. TDES1[25] takes precedence over TDES1[24].</p> |
| 23 | <p>DP: Disable Padding</p> <p>When set, the 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.</p> |
| 22 | Reserved. |
| 21:11 | <p>TBS2: Transmit Buffer 2 Size</p> <p>These bits indicate the Second Data Buffer in bytes. This field is not valid if TDES1[24] is set.</p> |
| 10:0 | <p>TBS1: Transmit Buffer 1 Size</p> <p>These bits indicate the First Data Buffer byte size. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of TCH (Bit 24).</p> |

Transmit Descriptor 2 (TDES2)

TDES2 contains the address pointer to the first buffer of the descriptor.

Table 10-15 Transmit Descriptor 2

| Bit | Description |
|------|--|
| 31:0 | <p>Buffer 1 Address Pointer</p> <p>These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment.</p> |

Transmit Descriptor 3 (TDES3)

TDES3 contains the address pointer either to the second buffer of the descriptor or the next descriptor.

Table 10-16 Transmit Descriptor 3

| Bit | Description |
|------|---|
| 31:0 | <p>Buffer 2 Address Pointer (Next Descriptor Address)</p> <p>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.)</p> |

15.6.4 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.

1. 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.
2. 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).
3. Provide the MAC address registers (GMAC_MAC_ADDR0_HI and GMAC_MAC_ADDR0_LO).
4. If Hash filtering is enabled in your configuration, program the Hash filter register (GMAC_HASH_TAB_HI and GMAC_HASH_TAB_LO).
5. 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.
6. 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
7. Program the Interrupt Mask register bits, as required, and if applicable, for your configuration.
8. 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.
9. 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.

15.6.5 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].

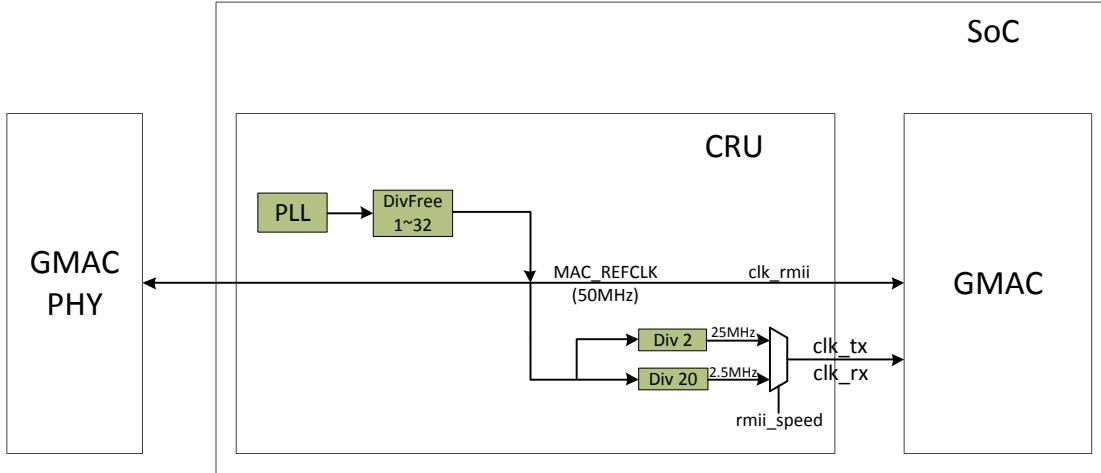


Fig. 10-48 RMII clock architecture when clock source from CRU

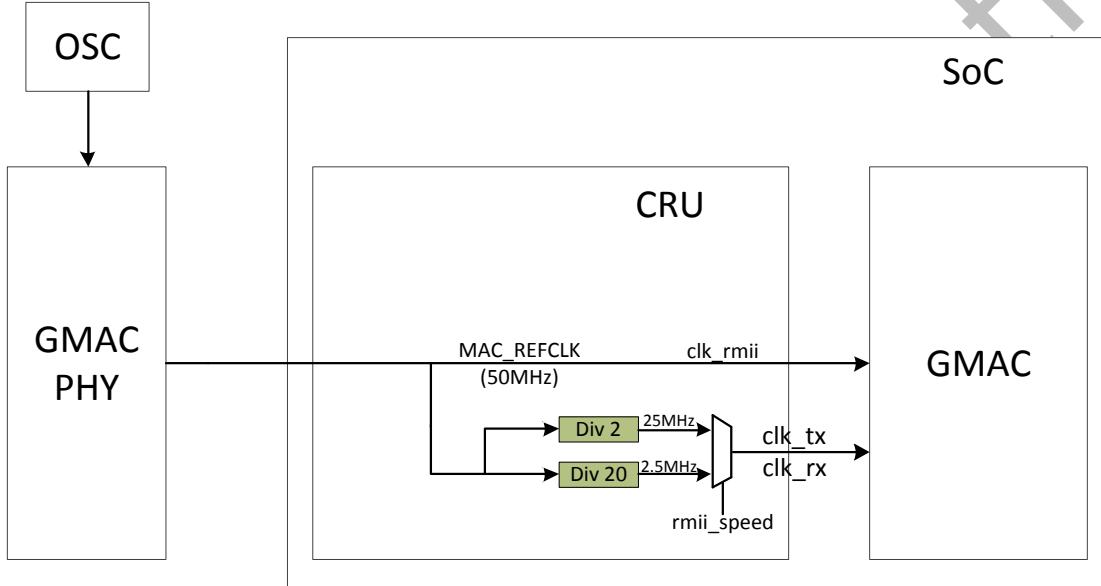


Fig. 10-49 RMII clock architecture when clock source from external OSC

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 clocks 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.

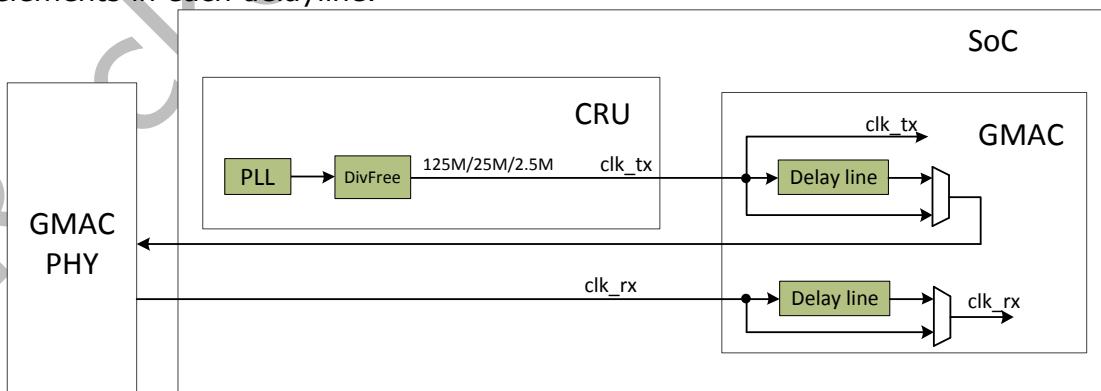


Fig. 10-50 RGMII clock architecture when clock source from CRU

15.6.6 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.

| | | | | | | | | | | | | |
|-------------------|--------------------|------------------|-----------------|------------------|-------------------|------------------|-----------------|------------------|--|--|--|--|
| wkupfmfilter_reg0 | Filter 0 Byte Mask | | | | | | | | | | | |
| wkupfmfilter_reg1 | Filter 1 Byte Mask | | | | | | | | | | | |
| wkupfmfilter_reg2 | Filter 2 Byte Mask | | | | | | | | | | | |
| wkupfmfilter_reg3 | Filter 3 Byte Mask | | | | | | | | | | | |
| wkupfmfilter_reg4 | RSVD | Filter 3 Command | RSVD | Filter 2 Command | RSVD | Filter 1 Command | RSVD | Filter 0 Command | | | | |
| wkupfmfilter_reg5 | Filter 3 Offset | | Filter 2 Offset | | Filter 1 Offset | | Filter 0 Offset | | | | | |
| wkupfmfilter_reg6 | Filter 1 CRC - 16 | | | | Filter 0 CRC - 16 | | | | | | | |
| wkupfmfilter_reg7 | Filter 3 CRC - 16 | | | | Filter 2 CRC - 16 | | | | | | | |

Fig. 10-51Wake-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.

15.6.7 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.

15.6.8 GRF Register Summary

| GRF Register | Register Description |
|--------------------|--|
| GRF_MAC_CON1[6:4] | PHY interface select 3'b001: RGMII 3'b100: RMII All others: Reserved |
| GRF_MAC_CON1[3] | 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_MAC_CON1[2] | GMACspeed 1'b1: 100-Mbps 1'b0: 10-Mbps |
| GRF_MAC_CON1[7] | RMII clock selection 1'b1: 25MHz 1'b0: 2.5MHz |
| GRF_MAC_CON1[9:8] | RGMII clock selection 2'b00: 125MHz 2'b11: 25MHz 2'b10: 2.5MHz |
| GRF_MAC_CON1[10] | RMII mode selection 1'b1: RMII mode 1'b0: Reserved |
| GRF_MAC_CON0[6:0] | RGMII TX clock delayline value |
| GRF_MAC_CON0[13:7] | RGMII RX clock delayline value |
| GRF_MAC_CON1[0] | RGMII TX clock delayline enable 1'b1: enable 1'b0: disable |
| GRF_MAC_CON1[1] | RGMII RX clock delayline enable 1'b1: enable 1'b0: disable |

Chapter 16 Smart Card Reader (SCR)

16.1 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

16.2 Block Diagram

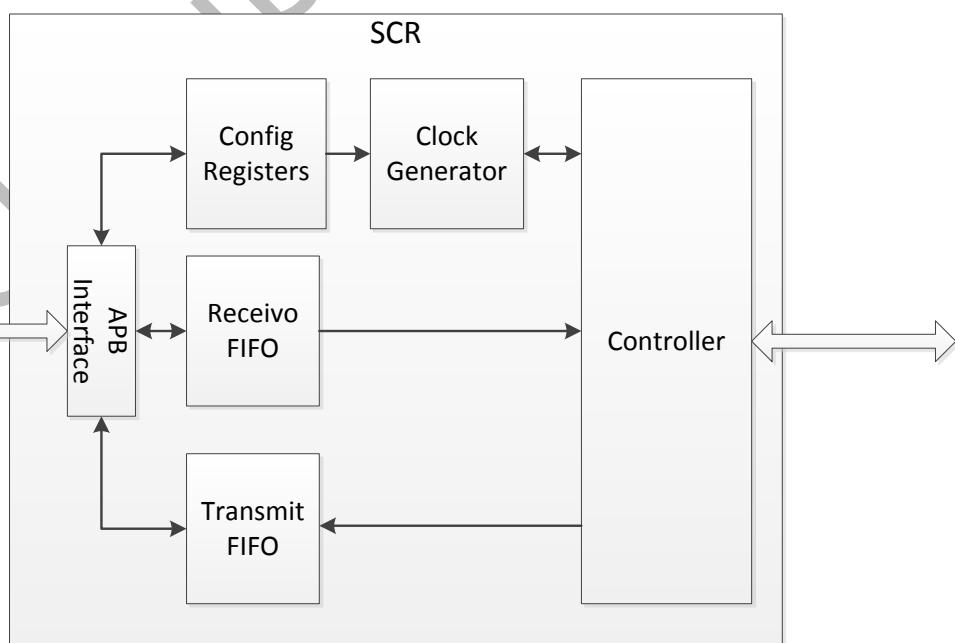


Fig. 10-52 SCR Block Diagram

The host processor gets access to PWM Register Block through the APB slave interface.

16.2.1 APB Interface

The host processor gets access to PWM Register Block through the APB slave interface.

16.2.2 Configuration Registers

The Configuration Registers block provides control over all functions of the Smart Card Reader

16.2.3 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 guardtime, parity checking and character repeating functions.

16.2.4 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.

16.2.5 Transmit FIFO

The Transmit FIFO is used to store the data to be transmitted to the Smart Card.

16.2.6 Clock Generator

The Clock Generator generates the Smart Card Clock signal and the Baud Clock Impulse signal, used in timing the Smart Card Reader.)

16.3 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

16.3.1 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.

16.3.2 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.

16.3.3 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 Vpp is necessary, the activation and deactivation part of Vpp 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. 10-53.

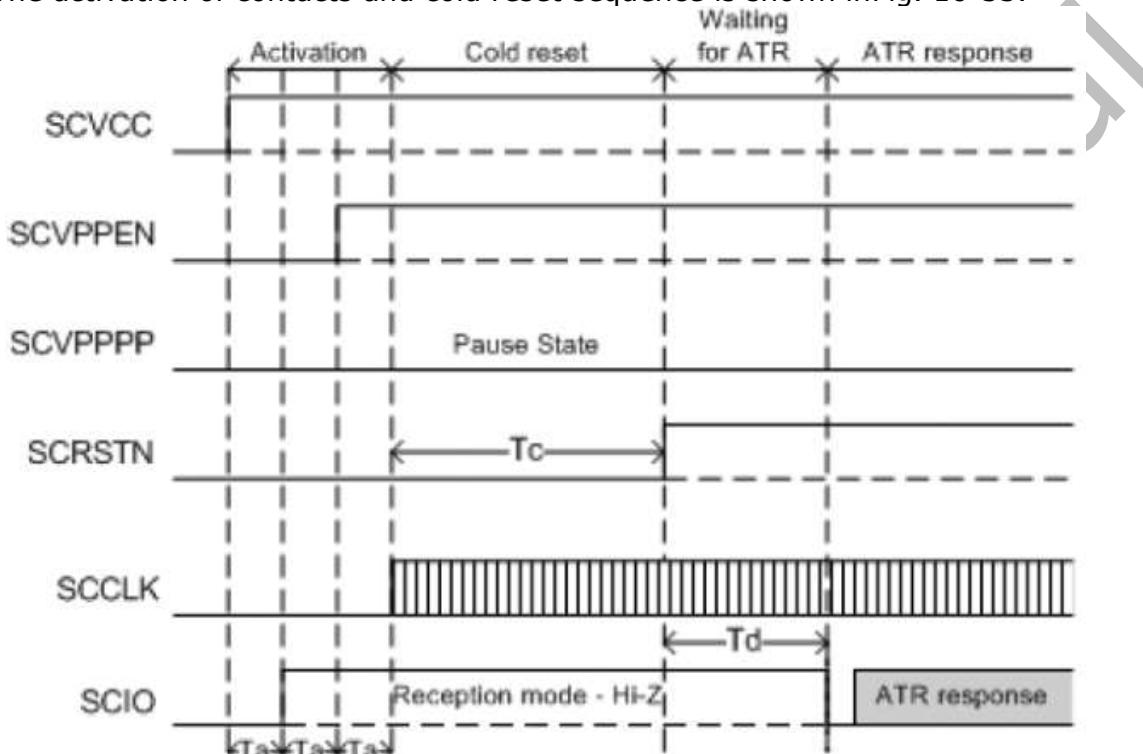


Fig. 10-53 Activation, Cold Reset and ATR

16.3.4 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.

16.3.5 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. 10-54. After the SCRSTN assertion, the Warm Reset sequence then continues the same way as the Cold Reset sequence.

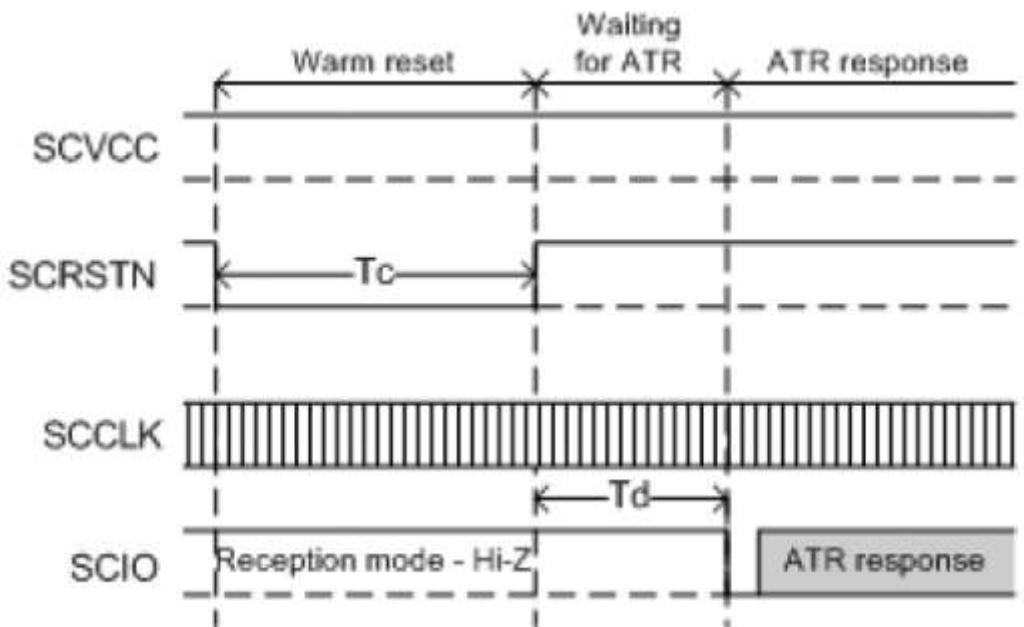


Fig. 10-54 Warm Reset and ATR

16.3.6 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.

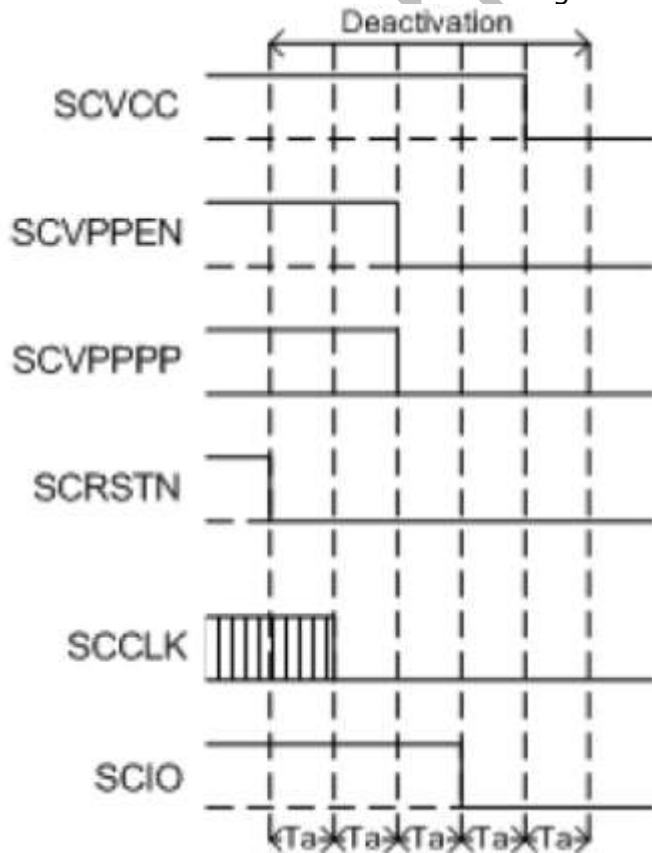


Fig. 10-55 Deactivation Sequence

16.4 Register Description

16.4.1 Registers Summary

| Name | Offset | Size | Reset Value | Description |
|------|--------|------|-------------|-------------|
|------|--------|------|-------------|-------------|

| 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_ATRSTARTLIMIT | 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

16.4.2 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 |
| 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. |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 11 | RW | 0x0 | <p>TOT1 T0/T1 Protocol Controls the using of T=0 or T=1 protocol. No character repeating is used when T=1 protocol is selected.</p> <p>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.</p> <p>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.</p> |
| 10 | RW | 0x0 | <p>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</p> |
| 9 | RW | 0x0 | <p>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.</p> |
| 8 | RW | 0x0 | <p>TXEN Transmission enable When enabled the characters are read from TX FIFO and transmitted through UART to the Smart Card</p> |
| 7 | RW | 0x0 | <p>CLKSTOPVAL Clock Stop Value The value of the scclk output during the clock stop state.</p> |
| 6 | RW | 0x0 | <p>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.</p> |
| 5:3 | RO | 0x0 | reserved |
| 2 | RW | 0x0 | <p>PECH2FIFO Character With Wrong Parity to FIFO Enables storage of the characters received with wrong parity in RX FIFO.</p> |

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 15:8 | RO | 0x00 | Reserved3 Reserved Reserved bits are hard-wired to zero |
| 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 if this class was selected. Default value after reset is |
| 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 if this class was selected. Default value after reset is |
| 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 'if this class was selected. Default value after reset is . |
| 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'to this bit initializes Warm Reset of the Smart Card. This bit is always read as. |
| 1:0 | RO | 0x0 | reserved |

SCR_SCPADS

Address: Operational Base + offset (0x0008)

Smart Card Pads Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 15:10 | RO | 0x0 | reserved |
| 9 | RO | 0x0 | SCPPRESENT Smart Card presented This bit is set to 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 =, 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 =, the DSCRST bit provides direct access to SCRST output |
| 2 | RW | 0x0 | DSCCLK Direct Smart Card Clock When DIRACCPADS =, the DSCCLK bit provides direct access to SCCLK output |
| 1 | RW | 0x0 | DSCIO Direct Smart Card Input/Output When DIRACCPADS =, 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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 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 |

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

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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. |
| 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. |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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_FIFOCTRL

Address: Operational Base + offset (0x0014)

FIFO Control Register

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 15:11 | RO | 0x0 | reserved |
| 10 | WO | 0x0 | RXFIFLUSH Flush RX FIFO RX FIFO is flushed, when it is written to this bit. |
| 9 | RO | 0x0 | RXFIFULL RX FIFO Full RX FIFO Full |
| 8 | RO | 0x0 | RXFIEMPTY RX FIFO Empty Field0000 Description |
| 7:3 | RO | 0x0 | reserved |
| 2 | WO | 0x0 | TXFIFLUSH Flush TX FIFO. TX FIFO is flushed, when it is written to this bit. |
| 1 | RO | 0x0 | TXFIFULL TX FIFO Full TX FIFO Full |

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 0 | RO | 0x0 | TXFIEMPTY TX FIFO Empty. TX FIFO Empty. |

SCR_LEGTXFICNT

Address: Operational Base + offset (0x0018)

Legacy TX FIFO Counter

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 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_LEGRXFICNT

Address: Operational Base + offset (0x0019)

Legacy RX FIFO Counter

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|---|
| 7:0 | RO | 0x00 | LEGRXFICNT 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

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

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

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

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

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

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 15:8 | RW | 0x00 | <p>ADEATIME Activation / Deactivation Time</p> <p>Sets the duration of each part of the activation and deactivation sequence. The value is in Smart Card Clock Cycles.</p> |

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|---|
| 7:0 | RW | 0x00 | Reserved Reserved Reserved bits are hard-wired to zero. |

SCR_LOWRSTTIME

Address: Operational Base + offset (0x0034)

Reset Duration

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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 ETUs. |

SCR_INTEN2

Address: Operational Base + offset (0x0040)

Interrupt Enable Register 2

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--------------------|
| 15:2 | RO | 0x0 | reserved |

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

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

| Bit | Attr | Reset Value | Description |
|------------|-------------|--------------------|--|
| 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_RXFIFOCNT

Address: Operational Base + offset (0x004c)

RX FIFO Counter

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

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

| Bit | Attr | Reset Value | Description |
|------|------|-------------|--|
| 15:0 | RO | 0x0000 | RXFIFO_CNT RX FIFO Counter This is a 16-bit, read-only register that provides the number of bytes stored in the RX FIFO. |

SCR_BAUDTUNE

Address: Operational Base + offset (0x0054)

Baud Tune Register

| Bit | Attr | Reset Value | Description |
|-----|------|-------------|--|
| 7:4 | RO | 0x0 | reserved |
| 3:0 | RW | 0x0 | BAUDTUNE 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

| 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. |

16.5 Interface Description

Table 10-17 NandC Interface Description

| Module Pin | Direction | Pad Name | IOMUX Setting |
|------------|-----------|------------------------------------|------------------------|
| sc_clk | O | IO_I2C2sda_GMACrx3_CARDclk_GPIO2c4 | GPIO2C_IOMUX[9:8]=11 |
| sc_RST | O | IO_I2C2scl_GMACrx2_CARDrst_GPIO2c5 | GPIO2C_IOMUX[11:10]=11 |
| sc_detect | I | IO_GMACtxd2_CARDdet_GPIO2c6 | GPIO2C_IOMUX[13:12]=10 |
| sc_io | I | IO_GMACtxd3_CARDio_GPIO2c7 | GPIO2C_IOMUX[15:14]=10 |

Notes: I=input, O=output, I/O=input/output, bidirectional

16.6 Application Notes**16.6.1 BCHST/BCHLOC/BCHDE/SPARE Application**

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:

$$SCCLK_{freq} = \frac{CLK_{freq}}{2 * (SCCDIV + 1)}, \quad SCCDIV \cong \frac{CLK_{freq}}{2 * SCCLK_{freq}} - 1$$

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 CardReader 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 ≥ 4 . The BAUD rate is given by the following equation:

$$BAUD_{rate} = \frac{CLK_{freq}}{2 * (BAUDDIV + 1)}$$

The duration of one bit, Elementary Time Unit (ETU) and parameters F and D are defined in the ISO/IEC7816-3 specification.

$$\frac{1}{BAUD_{rate}} \cong ETU = \frac{F}{D} * \frac{1}{SCCLK_{freq}}, \quad \frac{F}{D} \cong \frac{BAUDDIV + 1}{SCCDIV + 1}$$

BAUDDIV equation based on SCCDIV value and Smart Card parameters F and D is following:

$$BAUDDIV \cong (SCCDIV + 1) * \frac{F}{D} - 1$$

During the first answer to reset response after the cold reset, the initial ETU must be equal to 372 SmartCard Clock Cycles (given by parameters F=372 and D=1). In this case, the BAUDDIV should be:

$$BAUDDIV \cong (SCCDIV + 1) * \frac{372}{1} - 1$$

After the ATR is completed, the BAUDDIV register value can be changed according to Smart Cardparameters F and D.

Baud Tune Register (BAUDTUNE) 3-bit value that can be used to increase the accuracy of the BaudClock impulses timing by using the BAUDTUNE Increment from Table listed below in combination with BAUDDIVregister value.

Table 10-18BAUDTUNE register

| BAUDTUNE | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|--------------------------|-----|--------|------|--------|------|--------|-------|--------|
| BAUDTUNE _{INCR} | +0 | +0.125 | 0.25 | +0.375 | +0.5 | +0.625 | +0.75 | +0.875 |

$$BAUDDIV + BAUDTUNE_{INCR} \cong (SCCDIV + 1) * \frac{F}{D} - 1$$

The BAUDDIV register value (nearest integer) can be computed using following equation:

$$BAUDDIV \cong (SCCDIV + 1) * \frac{F}{D} - 1 - BAUDTUNE_{INCR}$$

Chapter 17 MACPHY

17.1 Overview

The OmniPhy 10 Base-T/100 Base-TX is a low-power Ethernet Physical Layer (PHY) transceiver with variable I/O voltage that is fully compliant with the IEEE 802.3 and 802.3u standards. The PHY works in both 10Base-T and 100Base-TX Fast Ethernet modes, supports 100Base-FX, and is fully compliant with the IEEE 802.3/802.3u standards. The core has register-selectable configuration-options, which define the functionality of the core.

It supports following features:

- Integrated IEEE 802.3/802.3u compliant 10/100Mbps Ethernet PHY
- Supporting both full and half duplex for either 10 or 100 Mb/s data rate
- Auto MDIX capable
- Supports wake-on-LAN, EEE
- 100Base-FX support
- RMII interface
- Supports auto-negotiation
- On-board diagnostics

17.2 Block Diagram

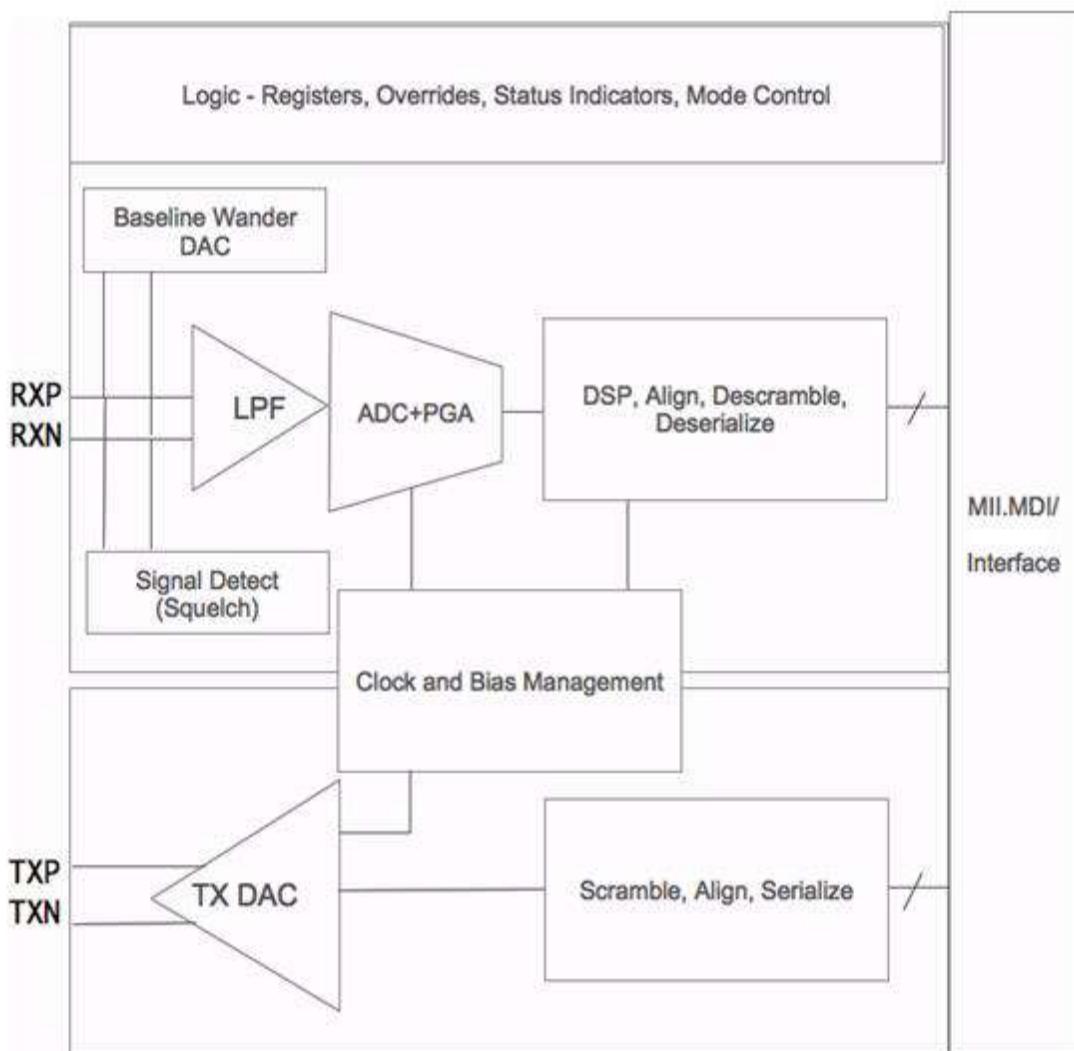


Fig. 10-56 AFE Block Diagram

17.3 Function Description

Functionally, the core PHY can be divided into the following sections:

- 100Base-TX transmit and receive
- 10Base-T transmit and receive
- Logic to handle auto-negotiation and management control

A block diagram of the analog-front-end is shown above. The functions of 100Base-T/FX, 10Base-T, and the related logic will be described thereafter. The complete set of power-modes provided by the standard are supported, including Energy-Detect and Idle modes for both 100- and 10BaseT modes.

100Base-TX and 100Base-TX Transmit and Receive

The transmitter is implemented using a Class-A current steering architecture. The transmitter drives a scrambled MLT3 data into the 100-ohm impedance. The transmitter implements a 6b DAC.

For the 100BaseT receive function, the MLT-3 from the cable is fed into the PHY through a low-pass-filter, and a 6b ADC samples the incoming data. A programmable gain is implemented in the ADC. Baseline wander is corrected using a small DAC.

The ADC outputs are then processed in the digital domain where they are equalized by a DSP block. The DSP also controls the recovered clock by selecting one of eight phases from the PLL output. This clock is used to recover the NRZI data-stream and descrambling takes place. The data are then aligned and decoded prior to being passed in 4b nibbles to the MAC through the MII interface (clocked at the controller rate).

Logic exists to detect if the receive data is valid and without errors (unexpected code groups within a frame).

In 100M FX mode, several sections of the signal path can be powered down and greatly simplified. For example, the scrambling/de-scrambling functions are disabled. However, a special signal detect is required to indicate the presence of an FX signal.

Fault detection monitoring is implemented according to Section 24.3.2.1 of the 802.3u standard.

Various over-ride and bypass mechanisms exist from within the register map

10Base-T Transmit and Receive

The transmitter serializes 2.5MHz nibbles (from the MAC) onto a 10M NRZ stream. This stream is then encoded using a Manchester encoding scheme and sent to the analog Class-A transmitter which drives the magnetics that ultimately drive the load. If no data is being sent, the link drives normal link pulses to the receiver, so that lock is not lost.

The receiver receives the encoded stream from the cable, and the analog signal is filtered and checked using a squelch circuit. The receiver recovers the clock and data to recreate the NRZI stream after confirming that the data is valid encoded data. Polarity is identified and corrected as necessary (observable through register interface). Then stream is deserialized ascent to the MAC interface at 2.5MHz.

Logic to handle auto-negotiation and management control

The PHY automatically determines the capabilities of the link partners according to the specification put forth in clause 28 of IEEE 802.3 standard. Information is passed back/forth using the SMI (serial management interface) bus and the results can be read via status registers. This is independent of the controller. The highest priority is given to 100M Full Duplex, then 100M Half Duplex, 10M Full Duplex, and 10M Half Duplex.

If the device the PHY is connected to lacks the capability, then a capability known as Parallel Detection will be employed – the speed will be determined by 100 MLT-3 Symbols or 10M Normal Link Pulses and that fact will be reflected in the management registers.

Auto-negotiation can be restarted at any time through a register setting. The normal transmission will be suspended for a time while the logic determines the right setting. Additionally, the PHY implements the Next Page function which allows the Auto-Negotiation infrastructure to transmit arbitrary information using acknowledge-based handshaking. Pages can either be predefined codes/instructions or unformatted bits.

Functional Modes

● Normal mode

In normal mode the PHY is fully operational.

When no packets are coming in and no packets need to be transmitted, the PHY is in Idle mode. The PHY must be in Idle mode for WOL detection.

● Energy Detect

In the Energy Detect mode the PHY is powered down, except for the register interface, the SQUELCH circuit and the ENERGYON logic. The ENERGYON logic is used to detect the presence of valid energy from 100Base-TX, 10Base-T or Auto-negotiation signals.

When energy is received (link pulses or packet) the PHY power on. It automatically resets itself into the state it had prior to power-down and asserts the interrupt output if enabled ("intreq"). The first and possibly the second packet to activate the PHY will be lost. Therefore this mode cannot be used for Wake-on-Lan detection. Energy Detect power down mode cannot be used when Auto-negotiation is enabled (Bit 0.12 must be cleared).

● PLL-Only

In the PLL-Only mode, only the PLL is active. This mode is required to have the 500 MHz clock from the PLL available in the system, while the rest of the PHY is not used (output "clk_out_500"). This clock can be used in a RGMII clocking scheme when the embedded PHY is bypassed and the MAC is operating in Gbps mode.

● True Power Down

In True Power Down mode the entire PHY is powered down except the SMI register interface. Access to the SMI registers is still possible, provided the external input clock ("clk_in") is active (25 MHz or 50 MHz). This is required in order to power up the PHY again via a register access.

The True Power Down mode is different from the General Power Down mode that is mentioned in the standard. This General Power Down mode corresponds to the PLL-Only mode, described in the table item above (please note that the standard does not say that the PLL is on in the so-called General Power Down mode).

● Disabled

In the Disabled mode the entire PHY is disabled. Even register access is not possible anymore.

● General Power-Down

This power-down is controlled by register. In this mode the entire PHY, except the management interface, is powered-down and stays in that condition as long as the relevant bit is held high. When it is cleared, the PHY powers up and is automatically reset. General power down mode requires external clock of 25/50MHz to be always present. In this case PLL can be shut off.

WOL (Wakeup on LAN)

The PHY should be operating in normal mode for WOL detection. The circuit monitors the network traffic for the special packets called the "Magic Packet". If the magic packet is received, it must further check to see if the magic packet's content has the correct information (CRC check on the received magic packet). If the magic packet contents are correct then the circuit issue an interrupt to the system host controller to wake up the reset of the circuitry and get the client out of the sleep mode.

A Magic packet is a frame addressed to the device, either a unicast to the programmed address or a broadcast with anywhere in a payload containing 48 '1's followed by 16 repetitions of the client's 48-bit MAC address, possibly followed by four or six byte password. The WoL design monitors all the packets transmitted from the Ethernet PHY and generates the WoL interrupt and the status for the following packet conditions.

- Detection of Magic Packet with Perfect DA
- Detection of Magic Packet with Broadcast DA
- Detection of only Magic Packet (Debug)

Refer to register specification for details on control.

The interrupt can be masked through the register 30 (Interrupt mask register) and interrupt status is available on register29 (interrupt status register).

MAC/Host controller should program the 48 bit MAC Address (Destination address) to the MAC address register.

Reset

- Power-Down reset:

Automatically activated when the PHY comes out of power-down mode. The internal power-down reset is extended by 256us after exiting the power-down mode to allow the PLL to stabilize before the logic is released from reset.

- Hardware reset

Connected to the reset_n input. If the reset_n input is driven by an external source, it should be held LOW for at least 100us to ensure that the core is properly reset.

- Software reset

Activated by writing a register. This signal is self-clearing. After the register-write, internal logic extends the reset by 256us to allow PLL-stabilization before releasing the logic from reset.

Rockchip Confidential