



# DesignWare Cores USB 2.0 Hi-Speed On-The-Go (OTG)

## Databook

---

*DWC\_otg – Product Codes*

## Copyright Notice and Proprietary Information

© 2020 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

### Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

### Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

### Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>

All other product or company names may be trademarks of their respective owners.

### Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

### Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.

[www.synopsys.com](https://www.synopsys.com)

# Contents

Revision History .....	13
<b>Preface .....</b>	<b>29</b>
Product Codes .....	29
Databook Organization .....	30
Related Product Documentation .....	31
Third-Party and Reference Documentation .....	32
Web Resources .....	32
Customer Support .....	32
<b>Chapter 1</b>	
<b>Product Overview .....</b>	<b>35</b>
1.1 System Overview .....	37
1.1.1 System-Level Block Diagram .....	37
1.1.2 Interfaces .....	38
1.1.3 Transmit and Receive FIFOs .....	39
1.2 Features List .....	39
1.2.1 General Features .....	40
1.2.2 Configurable Features .....	40
1.2.3 Application Interface Features .....	40
1.2.4 MAC-PHY Interface Features .....	42
1.2.5 System Memory Architecture .....	42
1.2.6 Non-DWORD Alignment Support .....	43
1.2.7 Internal Memory Features .....	43
1.2.8 Software Features .....	44
1.2.9 Power Optimization Features .....	44
1.2.10 Differences Between OTG Revision 1.3 and OTG Revision 2.0 .....	45
1.3 Changes from the Previous Release .....	46
1.4 Known Issues and Limitations .....	47
1.4.1 Feature and Verification Limitations .....	48
1.4.2 Worst-Case Inter-Packet Gap and Maximum Number of Cascaded Hubs Supported by Device Controller .....	49
1.4.3 Full-Scale Simulation .....	50
1.4.4 Thresholding .....	50
1.4.5 Synthesis .....	50
1.4.6 coreConsultant Configuration Parameters .....	51
1.4.7 DRC Violations .....	51
1.4.8 System-Level Testing .....	52
1.4.9 Software .....	53

1.4.10 Hardware Testing (Slave and Internal DMA) .....	53
1.4.11 MemoryMap .....	53
1.4.12 Limitations in the VTB .....	54
1.4.13 Spyglass .....	54
1.5 Compliance With Standards .....	55
1.6 Certification .....	56
1.7 Coding and Design Guidelines and Exceptions .....	56
1.8 Speed and Clock Requirements .....	57
1.9 Power and Area .....	58

## Chapter 2

Architecture .....	59
2.1 System Level Architecture .....	60
2.1.1 DWC_otg PMU Module .....	61
2.1.2 Design Hierarchy .....	63
2.1.3 DWC_otg Components .....	64
2.1.4 Host Architecture .....	65
2.1.5 Device Architecture .....	65
2.2 DWC_otg_core Architecture .....	69
2.2.1 AHB Bus Interface Unit (BIU) .....	71
2.2.2 Control and Status Registers (CSR) .....	80
2.2.3 Application Interface Unit (AIU) .....	81
2.2.4 Packet FIFO Controller (PFC) .....	84
2.2.5 Media Access Controller .....	92
2.2.6 PHY Interface Unit (PIU) .....	95
2.2.7 Wakeup and Power Controller (WPC) .....	96
2.2.8 List Processor .....	97
2.3 Power Management Unit Architecture .....	98
2.3.1 PMU Slave .....	99
2.3.2 PMU Logic .....	99
2.3.3 PMU IfMux Module .....	100
2.3.4 PMU Sync Module .....	101
2.3.5 ADP Module .....	102
2.4 System Clocks and Reset .....	103
2.4.1 System Clock Speeds .....	104
2.4.2 System Clock and Reset Generation .....	112
2.4.3 Clock Gating and Clock Tree Insertion .....	116
2.4.4 System Resets .....	117
2.4.5 Clock and Reset Distribution Without PMU .....	120
2.4.6 Clock and Reset Distribution With PMU .....	121
2.4.7 Clock and Reset Distribution With Extended Hibernation .....	122
2.4.8 Clock Domain Crossing .....	123
2.5 Interface and Protocol Timing .....	128
2.5.1 Control Write .....	128
2.5.2 Device Mode Bulk OUT .....	131
2.5.3 Device Mode Bulk IN .....	132
2.5.4 Device Mode Interrupt OUT .....	133
2.5.5 Device Mode Isochronous IN .....	134
2.5.6 Host Mode Isochronous IN .....	136

2.5.7 Host Mode Bulk Out in Slave Mode .....	138
2.6 USB 1.1 FS Transceiver Interface .....	139
2.7 Embedded Host with Micro-AB Receptacle Support .....	139
<b>Chapter 3</b>	
Configuration Parameters .....	141
3.1 Basic Config Parameters .....	142
3.2 USB Physical Layer Interface Parameters .....	146
3.3 Device Endpoint Configuration Parameters .....	150
3.4 Host Endpoint Configuration Parameters .....	152
3.5 Endpoint Channel FIFO Configuration Parameters .....	153
3.6 Additional Configuration Options Parameters .....	158
3.7 Endpoint Direction Parameters .....	164
3.8 Device Periodic FIFO Depth Parameters .....	165
3.9 Device IN Endpoint FIFO Depth Parameters .....	166
3.10 Recommended Configuration Parameter Values .....	167
<b>Chapter 4</b>	
Signal Interfaces .....	169
4.1 Signal Interfaces and Naming Conventions .....	169
4.2 I/O Delays .....	170
4.3 Device Wire Adaptor Signals .....	171
4.4 Debug Ports .....	172
4.4.1 Debug Probes in AHB Clock Domain (internal_probes) .....	172
4.4.2 Debug Probes in PHY Clock Domain (internal_probes_p) .....	174
4.5 Overriding Control of PHY Voltage Valid and ID Input Signals .....	179
<b>Chapter 5</b>	
Signal Descriptions .....	181
5.1 Common Clock and Reset Signals .....	184
5.2 AHB Slave Interface Signals .....	189
5.3 AHB Master Interface Signals .....	194
5.4 External DMA Controller Interface Signals .....	199
5.5 External Data FIFO (DFIFO) RAM Interface Signals .....	202
5.6 UTMI+ Parallel Interface Signals .....	205
5.7 UTMI+ PMU Interface Signals .....	228
5.8 UTMI+ OTG Interface Signals .....	231
5.9 UTMI+ Vendor Control Interface Signals .....	242
5.10 ULPI Interface Signals .....	244
5.11 I2C Interface Signals .....	249
5.12 LPM Interface Signals .....	251
5.13 HSIC Interface Signals .....	254
5.14 IC_USB Interface Signals .....	255
5.15 Simulation Control Interface Signals .....	267
5.16 Miscellaneous Interface Signals .....	268
5.17 Remote Memory Support Interface Signals .....	275
5.18 ADP Interface Signals .....	278
5.19 Battery Charger ACA Interface Signals .....	280
<b>Chapter 6</b>	

Control and Status Registers .....	283
6.1 Control and Status Registers Overview .....	283
6.2 CSR Memory Map .....	283
6.2.1 Interrupt Hierarchy .....	286
6.3 Overview of Commonly Used Register Bits .....	289
6.4 Additional Details on GLPMCFG.HIRD_Thres Register Field .....	295
 Chapter 7	
Register Descriptions .....	297
7.1 DWC_otg_map/DWC_otg_intreg Registers .....	302
7.1.1 GOTGCTL .....	307
7.1.2 GOTGINT .....	319
7.1.3 GAHBCFG .....	323
7.1.4 GUSBCFG .....	332
7.1.5 GRSTCTL .....	346
7.1.6 GINTSTS .....	355
7.1.7 GINTMSK .....	371
7.1.8 GRXSTSR .....	379
7.1.9 GRXSTSP .....	384
7.1.10 GRXFSIZ .....	388
7.1.11 GNPTXFSIZ .....	389
7.1.12 GNPTXSTS .....	392
7.1.13 GI2CCTL .....	395
7.1.14 GPVNDCTL .....	399
7.1.15 GPIO .....	402
7.1.16 GUID .....	403
7.1.17 GSNPSID .....	404
7.1.18 GHWCFG1 .....	405
7.1.19 GHWCFG2 .....	406
7.1.20 GHWCFG3 .....	414
7.1.21 GHWCFG4 .....	420
7.1.22 GLPMCFG .....	429
7.1.23 GPWRDN .....	445
7.1.24 GDFIFO CFG .....	456
7.1.25 GADPCTL .....	457
7.1.26 GREFCLK .....	463
7.1.27 GINTMSK2 .....	465
7.1.28 GINTSTS2 .....	466
7.1.29 HPTXFSIZ .....	467
7.1.30 DPTXFSIZ <sub>i</sub> (for i = 1; i <= OTG_NUM_PERIO_EPS-1) .....	469
7.1.31 DIEPTXF <sub>i</sub> (for i = 1; i <= OTG_NUM_IN_EPS-1) .....	471
7.1.32 HCFG .....	474
7.1.33 HFIR .....	481
7.1.34 HFNUM .....	483
7.1.35 HPTXSTS .....	485
7.1.36 HAINT .....	488
7.1.37 HAINTMSK .....	489
7.1.38 HFLBAddr .....	490
7.1.39 HPRT .....	491

7.1.40 HCCHARi (for i = 0; i <= OTG_NUM_HOST_CHAN-1)	500
7.1.41 HCSPLTi (for i = 0; i <= OTG_NUM_HOST_CHAN-1)	506
7.1.42 HCINTi (for i = 0; i <= OTG_NUM_HOST_CHAN-1)	509
7.1.43 HCINTMSKi (for i = 0; i <= OTG_NUM_HOST_CHAN-1)	515
7.1.44 HCTSIZi (for i = 0; i <= OTG_NUM_HOST_CHAN-1)	519
7.1.45 HCDMAi (for i = 0; i <= OTG_NUM_HOST_CHAN-1)	523
7.1.46 HCDMABi (for i = 0; i <= OTG_NUM_HOST_CHAN-1)	526
7.1.47 DCFG	527
7.1.48 DCTL	534
7.1.49 DSTS	544
7.1.50 DIEPMSK	547
7.1.51 DOEPMSK	550
7.1.52 DAINT	554
7.1.53 DAINTMSK	560
7.1.54 DTKNQR1	566
7.1.55 DTKNQR2	568
7.1.56 DVBUUSDIS	569
7.1.57 DVBUSPULSE	570
7.1.58 DTHRCTL	571
7.1.59 DTKNQR3	575
7.1.60 DIEPEMPMSK	576
7.1.61 DTKNQR4	578
7.1.62 DEACHINT	579
7.1.63 DEACHINTMSK	582
7.1.64 DIEPEACHMSK0	588
7.1.65 DIEPEACHMSKi (for i = 1; i <= OTG_NUM_EPS)	591
7.1.66 DOEPEACHMSK0	594
7.1.67 DOEPEACHMSKi (for i = 1; i <= OTG_NUM_EPS)	598
7.1.68 DIEPCTL0	602
7.1.69 DIEPINT0	608
7.1.70 DIEPTSIZ0	614
7.1.71 DIEPDMA0	616
7.1.72 DTXFSTS0	617
7.1.73 DIEPDMA0	619
7.1.74 DIEPCTLi (for i = 1; i <= OTG_NUM_EPS)	620
7.1.75 DIEPINTi (for i = 1; i <= OTG_NUM_EPS)	630
7.1.76 DIEPTSIZi (for i = 1; i <= OTG_NUM_EPS)	636
7.1.77 DIEPDMAi (for i = 1; i <= OTG_NUM_EPS)	638
7.1.78 DTXFSTS <sub>i</sub> (for i = 1; i <= OTG_NUM_EPS)	639
7.1.79 DIEPDMABi (for i = 1; i <= OTG_NUM_EPS)	641
7.1.80 DOEPCCTL0	642
7.1.81 DOEPPINT0	647
7.1.82 DOEPTSIZ0	653
7.1.83 DOEPPDMA0	655
7.1.84 DOEPPDMA0	656
7.1.85 DOEPCCTLi (for i = 1; i <= OTG_NUM_EPS)	657
7.1.86 DOEPPINTi (for i = 1; i <= OTG_NUM_EPS)	666
7.1.87 DOEPTSIZi (for i = 1; i <= OTG_NUM_EPS)	672
7.1.88 DOEPPDMAi (for i = 1; i <= OTG_NUM_EPS)	674

7.1.89	DOEPDMABi (for i = 1; i <= OTG_NUM_EPS)	675
7.1.90	PCGCCTL	676
7.1.91	PCGCCTL1	684
 Chapter 8		
Unified Power Format Support		687
8.1	Overview	687
8.2	Power Clamps	688
8.3	Power Switch Implementation Based on Polarity	688
8.4	UPF Support for the DWC_otg Controller	689
8.4.1	Requirements for UPF-based Power Aware Verification and Synthesis	689
8.4.2	UPF-Related Files in the HS OTG Controller	689
8.5	DWC_otg Hibernation Power Domains	690
8.6	DWC_otg Extended Hibernation Power Domains	691
 Appendix A		
Example Transfers in		
Device Scatter/Gather DMA Mode		695
A.1	Example Transfer: Bulk IN	696
A.2	Example Transfer: Bulk OUT	697
A.3	Example Transfer: Isochronous IN	698
 Appendix B		
Area, Speed, and Power		699
B.1	Area	699
B.2	PMU Area Numbers	709
B.3	Estimating Area for Your Configuration	709
B.4	Speed	711
B.5	Power-Compiler Clock Gating	711
B.6	Power Consumption	712
 Appendix C		
DWC_otg LPM Add-On Feature		715
C.1	Device Mode	716
C.2	Host Mode	717
C.3	Device-Only LPM Functions	717
C.3.1	Support for Link Power Management State	717
C.3.2	Functions to Support L1 Entry	718
C.3.3	Transition from L0 to L1	721
C.3.4	L1 State Events	722
C.3.5	Device-Initiated L1 Exit	723
C.3.6	Host/Hub-Initiated L1 Exit	723
C.3.7	Reset Signaling-Initiated L1 Exit	724
C.3.8	Host/Hub-Initiated L1 Exit with EnBESL = 1'b1	724
C.3.9	Reset Signaling-Initiated L1 Exit with EnBESL = 1'b1	725
C.4	Host-Only Functions	726
C.4.1	Communication of LPM Transactions to Devices	726
C.4.2	Entering the Sleep State	726
C.4.3	Entering the L1 State	727
C.4.4	L1 State Events	728

C.4.5 Device-Initiated L1 Exit .....	729	
C.4.6 Host-Initiated L1 Exit .....	729	
C.5 Implementing LPM Over ULPI Interface .....	730	
 Appendix D		
Active Clock Gating		
Add-On Feature .....	731	
D.1 Active Clock Gating Feature .....	732	
D.1.1 AHB and PHY Clock Gating .....	732	
D.1.2 RAM Clock Gating .....	735	
D.2 Configuring DWC_otg Controller for Active Clock Gating .....	736	
D.3 Programming DWC_otg Controller for Active Clock Gating .....	736	
 Appendix E		
Service Interval-Based Scheduling for ISOC IN Endpoints .....		737
E.1 Service Interval Mode Feature .....	738	
E.1.1 Using ref_clk for ADC 3.0 Support .....	739	
E.1.2 Auto Remote Wakeup Feature .....	739	
E.2 Configuring DWC_otg Controller for Using Service Interval-Based Scheduling for Hardware .....	740	
E.3 Programming Configurations Available with Service Interval Mode Enabled Configuration .....	740	
 Appendix F		
DWC_otg HSIC Add-On Feature .....		741
F.1 Common Functions in Device-Only and Host-Only Cores .....	741	
F.2 Detailed Device-Only Functions .....	742	
F.3 Detailed Host-Only Functions .....	742	
F.4 UTMI Signaling Details .....	743	
F.4.1 HSIC Attach and Enumeration .....	743	
F.4.2 Compile Time Configuration Options .....	744	
F.4.3 Host-Initiated Resume .....	745	
F.4.4 Device-Initiated Remote Wakeup .....	746	
 Appendix G		
DWC_otg IC_USB Add-On Feature .....		747
G.1 Detailed Device-Only Functions .....	747	
G.1.1 Functions to Support Self- and Bus-Powered Peripherals .....	747	
G.1.2 Functions to Support Removable and Fixed Peripherals .....	747	
G.1.3 Multiple Voltage Class .....	748	
G.1.4 Control of SW3, SW4, SW5 and SW6 in LS Mode of Operation .....	748	
G.1.5 Control of SW3, SW4, SW5 and SW6 in FS Mode of Operation .....	750	
G.1.6 Conversion of FS_LS Signals to IC_USB Signals .....	751	
G.2 Detailed Host-Only Functions .....	752	
G.2.1 Removable Peripheral or Fixed Peripheral Option .....	752	
G.2.2 Support for Both Self- or Bus-Powered Peripheral .....	752	
G.2.3 Multiple Voltage Class .....	752	
G.2.4 Control of SW1 and SW2 in FS and LS Modes of Operation .....	753	
G.2.5 Conversion of FS_LS Signals to IC_USB Signals .....	753	
 Appendix H		
Remote Memory Support for Internal DMA Configurations .....		755

H.1 Last Transaction in Remote Memory Support .....	759
H.2 Requirements .....	759
H.3 Mode of Operation .....	759
<b>Appendix I</b>	
DWC_otg Hibernation Add-on Feature .....	761
I.1 Overview of the Hibernation Sequence .....	761
I.1.1 DWC_otg Controller Functionality for Hibernation .....	762
I.1.2 Signal Changes to Support Hibernation .....	764
I.1.3 Scenarios Illustrating DWC_otg Controller Behavior When PHY Clock is On when Controller is in Off State (PMU Active) .....	766
<b>Appendix J</b>	
UTMI-to-UTMI Bridge Add-on Feature .....	769
J.1 Overview of U2U Bridge .....	770
J.1.1 Introduction .....	770
J.1.2 U2U Bridge Functions .....	770
J.1.3 Controller Requirements .....	772
J.1.4 U2U Bridge - System-Level Block Diagram .....	773
J.2 U2U Bridge Operational Sequences .....	776
J.2.1 Connect Sequence .....	776
J.2.2 Host Reset Sequence Following Initial Device Connect .....	778
J.2.3 Host Reset Sequence During Normal Traffic .....	779
J.2.4 Normal Traffic Flow in FS Idle Following Host Reset .....	779
J.2.5 Suspend Sequence .....	781
J.2.6 Device-Initiated Remote Wakeup Sequence .....	783
J.2.7 Host-Initiated Resume Sequence .....	785
J.2.8 Host-Initiated Resume Sequence in Parallel Mode .....	787
J.2.9 Host Reset During Suspend Sequence .....	788
J.2.10 Disconnect Sequence .....	790
J.3 U2U Bridge Clock Logic .....	791
J.3.1 Clock Request/Ack .....	791
J.3.2 u2u_dev_utmi_clk .....	791
J.3.3 u2u_host_utmi_clk .....	791
J.3.4 Internal Clocks .....	792
J.4 U2U Bridge Reset Logic .....	793
J.4.1 Power-on Reset Sequence .....	793
J.4.2 PMC_DEEP_RST_B .....	794
J.4.3 dev_u2u_utmi_reset .....	794
J.4.4 host_u2u_utmi_reset .....	795
J.5 Configuration Parameter .....	796
J.6 Signal Descriptions .....	797
J.6.1 U2U Bridge: Device Controller Interface Descriptions Signals .....	797
J.6.2 U2U Bridge: Host Controller Interface Descriptions Signals .....	801
J.6.3 U2U Bridge: System Interface Descriptions Signals .....	806
J.6.4 U2U Bridge: Debug Port Interface Descriptions Signals .....	811
J.6.5 16-Bit Data Interface .....	827
J.7 Registers .....	827
J.8 Area .....	827

J.9 References .....	828
J.10 Integrating DWC_U2UB Device Interface with DWC_otg Controller in Non-OTG Device Mode .....	829
Appendix K	
Chirp_on Behavior .....	831
K.1 FS Idle to Host Reset .....	832
K.2 FS Suspend to Host Reset .....	832
K.3 HS Idle to Host Reset .....	833
K.4 HS Idle to Host Suspend to Host Reset .....	834
K.5 HS IDLE to HS LPM Suspend to Host Reset .....	835
Appendix L	
Internal Parameter Descriptions .....	837



# Revision History



**Note** References to the sections, tables, figures, and page numbers in the Revision History table are only assured to be valid for the current version in the table.

Date	Version	Description
July 2020	4.20a	<ul style="list-style-type: none"> <li>■ Added           <ul style="list-style-type: none"> <li>- <a href="#">1.4.13</a> on page <a href="#">54</a></li> <li>- Footnote to the options that are no longer supported</li> </ul> </li> <li>■ Updated           <ul style="list-style-type: none"> <li>- Note on page <a href="#">36</a></li> <li>- <a href="#">1.2</a> on page <a href="#">39</a></li> <li>- <a href="#">1.3</a> on page <a href="#">46</a></li> <li>- <a href="#">1.4.1</a> on page <a href="#">48</a></li> <li>- <a href="#">1.4.8</a> on page <a href="#">52</a></li> <li>- <a href="#">1.4.12</a> on page <a href="#">54</a></li> <li>- <a href="#">2.2.1.1</a> on page <a href="#">71</a></li> <li>- <a href="#">Figure 2-32</a> on page <a href="#">113</a></li> <li>- <a href="#">Figure 2-34</a> on page <a href="#">115</a></li> <li>- <a href="#">Table 4-1</a> on page <a href="#">172</a></li> <li>- <a href="#">Table 4-2</a> on page <a href="#">174</a></li> <li>- <a href="#">8.2</a> on page <a href="#">688</a></li> <li>- <a href="#">8.4</a> on page <a href="#">689</a></li> <li>- <a href="#">B.1</a> on page <a href="#">699</a></li> <li>- <a href="#">D.1.1.2</a> on page <a href="#">734</a></li> <li>- <a href="#">Figure D-2</a> on page <a href="#">734</a></li> <li>- <a href="#">Table E-1</a> on page <a href="#">740</a></li> <li>- Chapter 3, “Configuration Parameters”</li> <li>- Chapter 5, “Signal Descriptions”</li> <li>- Chapter 7, “Register Descriptions”</li> <li>- Appendix L, “Internal Parameter Descriptions”</li> </ul> </li> </ul>
February 2019	4.11a	<ul style="list-style-type: none"> <li>■ Updated:           <ul style="list-style-type: none"> <li>- Chapter 3, “Configuration Parameters”</li> <li>- Chapter 4, “Signal Descriptions”</li> <li>- Chapter 5, “Control and Status Registers”</li> <li>- Chapter L, “Internal Parameter Descriptions”</li> </ul> </li> </ul>

Date	Version	Description
November 2018	4.10a	<ul style="list-style-type: none"> <li>■ Added:           <ul style="list-style-type: none"> <li>- ref_clk in “System Clocks and Reset” on page 102, Figure 2-38 on page 118, Figure 2-39 on page 119, Figure 2-40 on page 120, and “Clock Domain Crossing” on page 121</li> <li>- Table 2-3 on page 104</li> <li>- OTG_SERV_INT_ENH configuration parameter</li> <li>- “Recommended Configuration Parameter Values” on page 165</li> <li>- ref_clk signal</li> <li>- GREFCLK, GINTMSK2, and GINTSTS2 registers</li> <li>- Testmode_corr_eUSB2, ippisocSupt, ServIntFlow, EnhancedLPMSupt1, ServInt, StsPhsRcvdMsk, RAMGateEN register fields</li> <li>- “AHB Clock Frequency and AHB Latency Requirements” on page 111</li> <li>- Figure D-3 on page 721</li> <li>- “Service Interval-Based Scheduling for ISOC IN Endpoints” on page 723</li> </ul> </li> <li>■ Updated:           <ul style="list-style-type: none"> <li>- “Worst-Case Inter-Packet Gap and Maximum Number of Cascaded Hubs Supported by Device Controller” on page 48</li> <li>- “Impact of System Clocks on Minimum Inter-Packet Gap and Number of Cascaded Hubs Supported by the Device” on page 104</li> <li>- “Gate Simulation” section as “Identifying Metastability Paths in Gate-Level Simulation” on page 122</li> <li>- Registered, Power Domain, and Synchronous to attributes for signals</li> <li>- utmifs_dp_rpu1_en_n, utmifs_dp_rpu2_en_n, and utmifs_dm_pullup_en, signals</li> <li>- EOPF, I2CSuspCtl, LPM_RetryCnt_Sts, EnhancedLPMSupt, LPM_Retry_Cnt, LPM_Chnl_Idx, AppL1Res, and LPMCap register fields</li> <li>- “NYET Response” on page 706</li> <li>- “ACK Response” on page 707</li> <li>- “Limitations of Active Clock Gating Feature” on page 720</li> <li>- “RAM Clock Gating” on page 721</li> <li>- “Programming DWC_otg Controller for Active Clock Gating” on page 722</li> </ul> </li> </ul>

Date	Version	Description
April 2017	4.00a	<ul style="list-style-type: none"> <li>■ Added:           <ul style="list-style-type: none"> <li>- “Worst-Case Inter-Packet Gap and Maximum Number of Cascaded Hubs Supported by Device Controller” on page 48</li> <li>- Information on the number of cascaded hubs that the device controller can support in “Speed and Clock Requirements” on page 56</li> <li>- “Impact of System Clocks on Minimum Inter-Packet Gap and Number of Cascaded Hubs Supported by the Device” on page 104</li> <li>- hreset_gen_ahbm_n signal to “System Resets” on page 115, Figure 2-32 on page 111, Figure 2-34 on page 113, Figure 2-38 on page 118, Figure 2-39 on page 119, and Figure 2-40 on page 120</li> <li>- Note on AHB clock frequencies tested in “Clock Domain Crossing” on page 121</li> <li>- pwr_switch and test_bypass_lp input signals</li> <li>- “Debug Ports” on page 172</li> <li>- Note on reset value read requirement for the following registers/register fields: CurMod, BSesVld, ConIDSts, ConIDStsChng, CurMod, Host Periodic Transmit FIFO Size Register (HPTXFSIZ), and Host Frame Number/Frame Time Remaining Register (HFNUM)</li> <li>- EnhancedLPMSupt register field</li> <li>- Note (regarding DCTL.IgnrFrmNum bit) in SetOddFr and SetEvenFr register field descriptions</li> <li>- “Power Clamps” on page 674 and “Power Switch Implementation Based on Polarity” on page 674</li> <li>- Note on audio class support in the introduction of Appendix C, “DWC_otg LPM Add-On Feature”</li> <li>- Appendix D, “Active Clock Gating Add-On Feature”</li> </ul> </li> <li>■ Updated:           <ul style="list-style-type: none"> <li>- “Power Optimization Features” on page 43</li> <li>- “Known Issues and Limitations” on page 45</li> <li>- “Device Mode Functions” on page 93</li> <li>- “Data Width of the UTMII+ Interface” configuration parameter</li> <li>- sof_toggle_out signal (Active state as Toggle)</li> <li>- GRSTCTL.RxFFlsh (memory access to R_WS_SC)</li> <li>- INEPNakEff, CSftRst, and LPM_Retry_Cnt (register field descriptions)</li> <li>- “UPF Support for the DWC_otg Controller” on page 675</li> <li>- DOEPCTL0[20] (Snp) and DIEPCTLn/DOEPCTLn[20] (Snp) as Reserved</li> </ul> </li> <li>■ Removed “Instantiate Double-Synchronization Flops?” configuration parameter (OTG_INST_DSYNC_FLOPS)</li> </ul>

Date	Version	Description
October 2015	3.30a	<ul style="list-style-type: none"> <li>■ Added:           <ul style="list-style-type: none"> <li>- “Known Issues and Limitations” on page 45 (this section is duplicated from the Release Notes (DWC_otg_relnotes.pdf))</li> <li>- OTG_SYNC_RESET_TYPE configuration parameter: A note on synchronous reset style</li> <li>- DWC_U2UB_EN configuration parameter</li> <li>- internal_probes_p[207:0]</li> <li>- dis_tx_ipgap_dly_check to Host Configuration Register (HCFG)</li> <li>- More information on DMAAddr register field</li> <li>- Added a note on Enbl_L1Gating</li> <li>- Appendix J, “UTMI-to-UTMI Bridge Add-on Feature”</li> </ul> </li> <li>■ Updated:           <ul style="list-style-type: none"> <li>- “Clock Domain Crossing” on page 121</li> <li>- OTG_INST_DSYNC_FLOPS</li> <li>- Values of utmiotg_valid and utmisrp_bvalid signal, and added a note on HSIC mode</li> <li>- internal_probes size (now 58:0)</li> <li>- Encoding for HFIRRldCtrl register field</li> <li>- Values of FrInt register field</li> <li>- Figure F-1 on page 729 (and Step 3 on page 729)</li> </ul> </li> </ul>
October 2014	3.20a	Added a new register field (StsPhseStrt)
August 2013	3.20a	<p>Added the following register bits:</p> <ul style="list-style-type: none"> <li>■ XCVRDLY</li> <li>■ PIUFSSftRst</li> <li>■ ErraticIntMsk</li> <li>■ PTxFEMpLvl (existed in the core but had not been documented in earlier versions of this Databook)</li> <li>■ CurMod</li> </ul> <p>Added a note in Appendix C, “DWC_otg LPM Add-On Feature” stating that the core (acting as a Host) does not support hibernation or partial power-down mode when it is in L1 state.</p> <p>Modified Figure 2-32 on page 111</p> <p>Modified Figures K-1, K-2, and K-3 for scale-down value change</p>

Date	Version	Description
December 2012	3.10a	<p>The following Databook changes have been made for this version of the product:</p> <ul style="list-style-type: none"> <li>■ The Databook is now split into two documents: Databook and User Guide. Refer to the DesignWare Cores USB 2.0 Hi-Speed On-The-Go (OTG) User Guide for usage information such as configuration, integration, verification, and implementing UPF. The Databook retains reference information such as architecture, features, area and power, and other such reference information.</li> <li>■ Signal changes: <ul style="list-style-type: none"> <li>- Corrected a wrong signal name - dev_hird_rcvd_tgl to dev_hird_vld_tgl throughout the Databook</li> <li>- Added information that the utmiotg_iddig signal is also present in SRP capable Host mode</li> <li>- Added a new signal: dbnce_fltr_bypass</li> </ul> </li> <li>■ Register changes: <ul style="list-style-type: none"> <li>- RestoreMode bit in the Power and Clock Gating Control Register (PCGCCTL) was wrongly marked as Read-Only. It is now changed to R/W.</li> <li>- Added the following new register bits: DbnceFltrBypass in the Control and Status Register (GOTGCTL) EHEN in the Control and Status Register (GOTGCTL)</li> <li>- DeepSleepBESLReject in the Device Control Register (DCTL)</li> <li>- Modified the description of HIRD_Thres to include related information about DCTL.DeepSleepBESLReject information.</li> <li>- The description of the DevHNPEn, HstSetHNPEn, and HNPReq register bits now state that these bits are valid only when the core is HNP-capable</li> <li>- In the Device IN Endpoint Common Interrupt Mask Register (DIEPMSK), the reserved field is changed from 12:9 to 12:10</li> </ul> </li> <li>■ Added a new section: “Embedded Host with Micro-AB Receptacle Support” on page 137</li> <li>■ Modified Figure F-1 and the corresponding detailed description</li> </ul>

Date	Version	Description
April 2012	3.00a	<p>The following Databook changes have been made for this version of the product:</p> <ul style="list-style-type: none"> <li>■ A new extended hibernation feature is added. Following are the associated documentation changes: <ul style="list-style-type: none"> <li>- Added extended hibernation to the list of power optimization features in “Power Optimization Features” on page 43.</li> <li>- Modified Figures 2-5, 2-6, 2-8, 2-9, 2-24, and 2-25 to illustrate 2 new modules that are implemented in the DWC_otg core - DWC_otg_piu_pie and DWC_otg_piu_stup_store. Added Figures 2-34 and 2-40</li> <li>- Added the DWC_otg_piu_pie and DWC_otg_piu_stup_store modules in the file hierarchy in Table 2-1.</li> <li>- Added two new signals: piu_hreset_n and piu_prst_n</li> <li>- Modified OTG_EN_PWROPT configuration parameter to reflect the new value (OTG_EN_PWROPT=3) that is now available for extended hibernation.</li> <li>- The ExtndedHibernation register bit field is added to the User HW Config4 Register (GHWCFG4). The ExtndedHibernationSwitch, ExtndedHibernationClamp, and EnExtndedHibernation register bit fields are added to the Power and Clock Gating Control Register (PCGCCTL)</li> <li>- Added a note in “Area, Speed, and Power” on page 685 about the 5K increase in area when extended hibernation is enabled.</li> <li>- The DWC_otg_always_on module is now renamed to DWC_otg_wpc_slv. All previous Databook references to DWC_otg_always_on module have also been modified.</li> </ul> </li> <li>■ DWC_otg now supports AHB clock frequencies up to 270 MHz. This information is now updated in “Application Interface Features” on page 40.</li> <li>■ Added Table 4-7 to describe the UTMI+ PMU Interface signals and updated Figure 4-1 with these new signals.</li> <li>■ Added a “Preview” table for all registers in Chapter 6, “Control and Status Registers”</li> <li>■ Added a new section “Implementing LPM Over ULPI Interface” on page 716.</li> <li>■ Updated OTG_RX_DFIFO_DEPTH parameter in Table 3-1 on page 131.</li> <li>■ Updated Bit 24 in Table 5-9 on page 256; Bits 29 and 28 in Table 5-29 on page 312; Bits [12:8] and Bits[5:3] in Table 5-29 on page 312; Bits [23:22] in Table 5-56 on page 375.</li> <li>■ Added “Host/Hub-Initiated L1 Exit with EnBESL = 1'b1” on page 710 and “Reset Signaling-Initiated L1 Exit with EnBESL = 1'b1” on page 711.</li> <li>■ Added “DWC_otg Extended Hibernation Power Domains” on page 677.</li> </ul>

Date	Version	Description
July 2011	2.94a	<p>The following Databook changes have been made for this version of the product:</p> <ul style="list-style-type: none"> <li>■ Programming information has been removed from the Databook and moved into a separate Programming Guide</li> <li>■ Chapter 2, “Configuring, Verifying, and Synthesizing the DWC_otg Core” has been restructured and more usage information has been added.</li> <li>■ The coreConsultant parameters table is now moved into a new chapter, Chapter 3, “Configuration Parameters”.</li> <li>■ Modified “AHB Master Bus Interface Unit (BIUM)” on page 77 to include information about the new AHB Single Support feature.</li> <li>■ Added an important note about setting OTG_TRANS_COUNT_WIDTH parameter in coreConsultant when DWC_otg core is operating in Scatter/Gather DMA mode.</li> <li>■ Added 2 new register bits: <ul style="list-style-type: none"> <li>- AHBSingleSupport (AHBSingle) to support single transfers for the remaining data in a transfer when the DWC_otg core is operating in DMA mode.</li> <li>- Enable Continue on BNA (EnContOnBNA) to enable the DWC_otg core to continue on BNA for Bulk OUT endpoints.</li> </ul> </li> <li>■ The reset value of Soft Disconnect (SftDiscon) is changed to 1'b1, and this bit is not reseted by soft reset from this version.</li> <li>■ Updated area numbers for all tables in Appendix B, “Area, Speed, and Power” and added a new table for area numbers using 40lp libraries (Table B-2).</li> <li>■ Bit 30:27 in the Host Configuration Register (HCFG) was wrongly listed as Bit 31:27. This defect is now fixed in the Databook.</li> </ul>
April 2011	2.93b	<p>This is a documentation only release. In this release, the following add-on components have been added to the cover page:</p> <ul style="list-style-type: none"> <li>■ 6922-0 DWC USB HSOTG Hibernate Add-On</li> <li>■ 4740-0 DWC USB HSOTG HSIC-LPM Add-on</li> <li>■ 4737-0 DWC USB 2.0 HS OTG-FS Only-ICUSB Add-on</li> </ul>
December 2010	2.93a	<p>The following documentation changes have been made for this version of the product:</p> <ul style="list-style-type: none"> <li>■ Added Host connect and disconnect flows in “Host Connection” on page 347 and “Host Disconnection” on page 348</li> <li>■ Added Device connect, disconnect, and soft disconnect flows in “Device Connection” on page 349, “Device Disconnection” on page 350, and “Device Soft Disconnection” on page 350</li> <li>■ Added a new programming flow - “Programming Model for Bulk OUT Endpoints With OUT NAK Set for Device Descriptor DMA Mode” on page 567</li> <li>■ Added a new programming flow - “Transfer Stop Process” on page 428</li> <li>■ Added a new chapter on power saving - “Power Management” on page 635. This chapter contains all the power-related programming flows such as LPM, Partial Power down, hibernation, and internal and external clock gating.</li> </ul>

Date	Version	Description
June 2010	2.92a	<p>The following documentation changes have been made for this version of the product:</p> <ul style="list-style-type: none"><li>■ Added chapter providing integration information called “Integrating the Core with the PHY, Application RTL, or Verification IP”.</li><li>■ Added information on overriding PHY voltage valid input signals in “Overriding Control of the PHY Voltage Valid and ID Input Signals” on page 180</li><li>■ Modified Figures 2-35, 2-36, 2-39, 2-44, 4-1, and I-2 to show the new hclk input clock port for PMU in Hibernation.</li><li>■ The following signal and register descriptions are modified:<ul style="list-style-type: none"><li>- “LPM Interface” on page 190</li><li>- “Miscellaneous Signal Interface” on page 197</li><li>- “Control and Status Register (GOTGCTL)” on page 246</li><li>- “Host Configuration Register (HCFG)” on page 338</li></ul></li><li>■ Added a new section - “Scenarios Illustrating DWC_otg Controller Behavior When PHY Clock is On when Controller is in Off State (PMU Active)” on page 752</li><li>■ Additional information has been added regarding Unified Power Format (UPF). All UPF-related information is now available in “Unified Power Format Support” on page 673Outdated signal, registers, and parameters information have been corrected.</li></ul>

Date	Version	Description
January 2010	2.91a	<p>Added the following features:</p> <ul style="list-style-type: none"> <li>■ Support for OTG Revision 2.0 programming model</li> <li>■ Support for Hibernation</li> <li>■ Support for Unified Power Format (UPF)</li> <li>■ Support for flexible FIFO sizes</li> </ul> <p>The documentation changes associated with these enhancements are as follows:</p> <ul style="list-style-type: none"> <li>■ Updated “Features List” on page 39</li> <li>■ Updated the coreConsultant configuration parameters in .</li> <li>■ Added the following sections in chapter 2: <ul style="list-style-type: none"> <li>- “DWC_otg PMU Module” on page 61</li> <li>- “The List Processor (LP) implements Descriptor-Based Scatter/Gather DMA for Device and Host mode.” on page 96</li> <li>- “Clock and Reset Distribution With PMU” on page 119</li> <li>- Reorganized the existing content and updated the existing architecture diagrams to illustrate the hibernation-related changes</li> </ul> </li> <li>■ Added Tables 4-19, and 4-20</li> <li>■ Updated Tables 5-7, 5-12 and added Tables 5-30, 5-32, and 5-81.</li> <li>■ Made the following changes in chapter 6: <ul style="list-style-type: none"> <li>- Added “OTG Revision 2.0 Programming Model” on page 582</li> <li>- Added “Hibernation Programming Model when the DWC_otg Core is in Host Mode” on page 655</li> </ul> </li> </ul> <p>Added the following new appendixes:</p> <ul style="list-style-type: none"> <li>■ “Wrapper Logic for Battery Charger Support”</li> <li>■ “Chirp_on Behavior” on page 817</li> </ul>
April 2009	2.90a	<p>Added the following features:</p> <ul style="list-style-type: none"> <li>■ Support for Host Scatter-Gather DMA mode,</li> <li>■ Support for an optional interface for Remote Memory Support used to signal the core of a DMA write complete event on the system</li> </ul> <p>The documentation changes associated with these enhancements are as follows:</p> <ul style="list-style-type: none"> <li>■ New section “Host Scatter-Gather DMA Mode” on page 500</li> <li>■ Updates to Table 3-1, Table 2-1, Table 2-1, Table 5-9, Table 5-28, Table 5-36, Table 5-44, Table 5-46, Table 5-47, Table 5-48, Table 5-49, Table 5-50, Table 5-51,</li> <li>■ Updated Figure 2-7, Figure 2-5, Figure 2-8, Figure 2-10, Figure 2-12, Figure 4-1,</li> <li>■ New section Section 4.4.15, “Remote Memory Support Signal Interface”</li> <li>■ Updated Section 6.17.1.2, “Host Mode” and Section 6.17.1.4, “Calculating the Total FIFO Size for DWC_otg Core”</li> <li>■ New appendix “Remote Memory Support for Internal DMA Configurations” on page 741</li> </ul>

Date	Version	Description
January, 2009	2.81a	<p>Added the following features:</p> <ul style="list-style-type: none"> <li>■ 32-KHz clock support for Device mode</li> <li>■ BIU master moved to Power-Down domain</li> <li>■ Two-threshold enhancement</li> <li>■ LPM support over ULPI</li> </ul> <p>Entered the following documentation changes:</p> <ul style="list-style-type: none"> <li>■ Removed Verilog testbench appendix</li> <li>■ Fixed utmifs_tx_dat definition in Table 4-9 on page 181</li> <li>■ Added DWC_otg-undef.v to Figure 2-1 on page 60</li> <li>■ Added AHB Threshold Ratio (AHBThrRatio) bus and modified Transmit Threshold Length (TxThrLen) description in Table 5-67 on page 385.</li> <li>■ Added information on the DTHRCTL.TxThrLen and DTHRCTL.AHBThrRatio register fields thresholding to “Thresholding in DMA Mode” on page 356.</li> <li>■ Changed register access characteristics for “Moving the Host Core to Test Mode” on page 350, Bit 30, Channel Disable (ChDis)</li> </ul>
October, 2008	2.80a	<ul style="list-style-type: none"> <li>■ Added LPM, HSIC, and IC_USB interfaces. <ul style="list-style-type: none"> <li>- UTMI+ PHY interface updated to operate with HSIC.</li> <li>- Full-speed transceiver interface updated to operate with IC_USB.</li> <li>- Configuration parameters added for LPM, HSIC, and IC_USB.</li> <li>- Added Appendix C, “DWC_otg LPM Add-On Feature”</li> <li>- Added Appendix F, “DWC_otg HSIC Add-On Feature”</li> <li>- Added Appendix G, “DWC_otg IC_USB Add-On Feature”</li> </ul> </li> <li>■ Added USB link power management option.</li> <li>■ Added sof_toggle_out signal to Miscellaneous Signal interface</li> <li>■ Added FS 32-KHz Suspend mode.</li> <li>■ Reordered Chapters 3 and 6.</li> <li>■ Updated compliance to On-the-Go 1.3 standard.</li> <li>■ Corrected Device Mode CSR Map (Table 5-3 on page 246)</li> <li>■ Corrected Port Reset (PrtRst) bit (Bit 8) of Host Port Control and Status Register: (HPRT) table</li> <li>■ Updated Figures 2-6, 2-5, and 2-8.</li> <li>■ Updated GHWCFG3[13] (OTG_ENABLE_HSIC)</li> </ul>

Date	Version	Description
August 29, 2008	2.72a	<p>Modified sections: 5.3.5.16, 5.3.5.17, 5.3.5.18, 5.3.5.19</p> <p>Updated:</p> <ul style="list-style-type: none"> <li>■ Table 5-10: Bit 28, TxEndDelay</li> <li>■ Table 5-24: Bits 31:0, to current core release number</li> <li>■ Table 5-26: Bit 20, MultiProcIntrpt</li> <li>■ Table 5-42: Bit 16, NakOnBble; and Bit 15, IgnrFrmNum</li> <li>■ Table 5-57: Bit 13, NAKMsk</li> <li>■ Table 5-58: Bit 14, NYETMsk; Bit 13, NAKMsk; Bit 12, BbleErrMsk</li> <li>■ Table 5-76: Bit 14, NYETIntrpt; Bit 13, NAKIntrpt; Bit 12, BbleErrIntrpt</li> </ul>
April, 2008	2.71a	<ul style="list-style-type: none"> <li>■ Added OUT-NAK enhancement to “Bulk and Control OUT/SETUP Transactions in DMA Mode” on page 315</li> <li>■ Added examples to the following sections <ul style="list-style-type: none"> <li>- Dedicated Slave</li> <li>- Bulk In (“Bulk and Control OUT/SETUP Transactions in Slave Mode” on page 310)</li> <li>- Isochronous In (“Isochronous IN Transactions in Slave Mode” on page 334)</li> <li>- Buffer Mode DMA Bulk In (“Bulk and Control IN Transactions in DMA Mode” on page 320)</li> </ul> </li> <li>■ Updated “B-Device Host Negotiation Protocol” on page 437</li> <li>■ Table 6-43, “Host Channel-n Transfer Size Register: HCTSIZn”Corrected field [24:23] in register Table 3-9, “IN Data Memory Structure Values”.</li> </ul>
February 15, 2008	2.70a	<p>This release adds support for Descriptor-based (Scatter/Gather) DMA mode.</p> <p>Added</p> <ul style="list-style-type: none"> <li>■ Sections 3.4, 4.3, 6.5, B.5, and Appendix E</li> </ul> <p>Updated</p> <ul style="list-style-type: none"> <li>■ Figures 3-2, 3-4, 3-5, 3-6, 3-8, 3-9, 3-10, 3-15, and 4-1</li> <li>■ Sections 5.3.5.1, 6.2.2, and 6.3</li> <li>■ Tables 1-1, 2-1, 3-1, 4-13, 5-3, 5-9, 5-43, 5-44, 5-47, 5-48, 5-59, 5-60, 5-61, 5-62, 5-65, 5-66</li> </ul> <p>Moved</p> <ul style="list-style-type: none"> <li>■ Building and verifying information from “Configuration Parameters” chapter to the <i>USB 2.0 Hi-Speed On-The-Go (OTG) Quickstart</i></li> </ul>
August 2007	2.66a	<ul style="list-style-type: none"> <li>■ Updated styles and formatting to current corporate standards</li> <li>■ Removed: DC-FPGA Support, example technology library</li> </ul>

Date	Version	Description
June 2007	2.65a	<p>Updated</p> <ul style="list-style-type: none"> <li>■ Tables <ul style="list-style-type: none"> <li>- 5-6 bit 19 (STAR 9000149156)</li> <li>- 5-7 bit 19 (STAR 9000149156)</li> <li>- 5-9 bits [30:29], [8] (STAR 9000149156)</li> <li>- 5-11 bit [29], [26], [5] (STAR 9000149156)</li> <li>- 5-19 bit [24] (STAR 9000149156)</li> <li>- 5-25 bit [31:26] (STAR 9000149156)</li> <li>- 5-26 bit [12] (STAR 9000149156)</li> <li>- 5-33 bits [31:16], [15:0] (STAR 9000149156)</li> <li>- 5-47 bit [2:1] (STAR 9000149156)</li> <li>- 5-63 bit [7] (STAR 9000149156)</li> <li>- 5-64 bit [20:19] (STAR 9000149156)</li> <li>- 5-66 bit [30:29] (STAR 9000149156)</li> </ul> </li> <li>■ Sections <ul style="list-style-type: none"> <li>- 1.3.3: AHB burst type and Busy cycles;</li> <li>- 3.3.4.1.1: bits [34:32]</li> <li>- 3.3.4.1.2: 3'b111 (STAR 9000149156)</li> <li>- 3.3.4.1.4: 3'b111 (STAR 9000149156) must clear GINTSTS (STAR 9000149153)</li> <li>- 6.5.1: Interrupt bit (STAR 9000149156)</li> <li>- 6.7.1.3.1: Host Non-Periodic TxFIFO and Device IN Endpoint-Specific TxFIFOs</li> <li>- 6.7.1.3.2: Dynamic FIFO Sizing Enabled</li> <li>- 6.7.1.3.3: Device IN Endpoint TxFIFO; B.1 PHY model bullet.</li> </ul> </li> <li>■ Figures <ul style="list-style-type: none"> <li>- 3-22, 3-25: Removed clk_stable; 3-41: hclk_gated</li> </ul> </li> <li>Added</li> <li>■ Note to 6.3.4.3.1</li> <li>■ Section 4.5</li> <li>■ “Running Simulations in ULPI and FS1.1 Transceiver.”</li> <li>Removed</li> <li>■ clk_stable from Figures 3-22, 3-25</li> </ul>
November 2006	2.60a	<ul style="list-style-type: none"> <li>■ Removed Outdated Demo Platform description.</li> <li>■ Updated Transmit and Receive FIFO information throughout.</li> <li>■ Updated Shared FIFO Operation throughout.</li> <li>■ Updated formatting to match current corporate guidelines.</li> <li>■ Updates: <ul style="list-style-type: none"> <li>- Sections: 1.2.3, 1.3, 1.6.3, 3.3.3.1, 4.4.2, 5.3.2.19, 5.3.2.20, 5.3.3.6, 5.3.4.14, 5.3.4.15, 5.3.4.22, 6.4.2, 6.4.2.1.5, 6.4.2.1.8, 6.4.2.2.1, 6.4.2.2.2, 6.2.4, 6.4.4, 6.7.1</li> <li>- Figures: 2-3, 3-8, 3-49, 4-1</li> <li>- Tables: 1-1, 4-2, 5-8, 5-9, 5-10, 5-28, 5-30, 5-44, 5-45, 5-48, 5-49, 5-58, 5-59, 5-63, 5-67, C-7</li> </ul> </li> </ul>

Date	Version	Description
August 2006	2.50a	<ul style="list-style-type: none"> <li>■ Updated Table 5-39, Host Channel-<i>n</i> Interrupt Register: HCINT<i>n</i>.</li> <li>■ Updated Figure 3-28, ULPI Carkit.</li> <li>■ Updated description for utmi_xcvrselect in Table 4-6, UTMI+ Parallel Interface.</li> <li>■ Updated Bulk/Control OUT code sample in Table 6-3, Interrupt Service Routines for Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in DMA Mode.</li> <li>■ Updated Section 1.3.2, Software Features.</li> <li>■ Updated Section 1.3.3, Application Features.</li> <li>■ Updated Table 1-1, DWC_otg Configurable Parameters.</li> <li>■ Updated signal internal_probes in Table 4-13, Miscellaneous Signal Interface Signals.</li> <li>■ Updated features in Table C-7, Areas for DWC_otg Features.</li> <li>■ Added Section 3.3.6.2.5, ULPI FsLs Serial mode.</li> <li>■ Updated bit description for Host Frame Number/Frame Time Remaining Register (HFNUM) [15:0].</li> <li>■ Updated register descriptions for sof_update_toggle and sof_count.</li> <li>■ Added description of Thresholding throughout databook</li> <li>■ Added description of Shared FIFO operation throughout databook.</li> <li>■ Copy edited previous publication for syntax and semantics.</li> </ul>

Date	Version	Description
March 2006	2.40a	<ul style="list-style-type: none"> <li>■ Updated reference to ULPI interface specification in Section 1.3.4.</li> <li>■ Updated Enable I<sup>2</sup>C Interface and Largest Rx Data FIFO Depth configuration parameters in Table 2-1.</li> <li>■ Added DWC_otg_piu_sync.v to Table 3-1.</li> <li>■ Updated clock descriptions in Section 3.5.1.</li> <li>■ Updated Figures 3-25 and 3-41, and added Figures 3-61 and 3-62.</li> <li>■ Updated ulpi_data_out_en bus in Figure 4-2 and corresponding description in Table 4-8.</li> <li>■ Updated fields 31, 30:23, and 21–17 in Table 5-9, field 0 in Table 5-10, field 22 in Table 5-11, field 0 in Table 5-45, field 6 in Table 5-49, and fields 31:6 in Table 5-63.</li> <li>■ Updated overview of Power and Clock Gating Control register in Section 5.3.5.1 and field 2 in Table 5-70.</li> <li>■ Updated Section 5.3.3.12, “Host Channel-n DMA Address Register (HCDMAn)” and Section 5.3.4.22, “Device Endpoint-n DMA Address Register (DIEPDMAn/DOEPDMAn).”</li> <li>■ Updated information about Periodic and Non-periodic Request Queue depths in Section 6.3.5.</li> <li>■ Updated description of application programming sequence in Section 6.2.4.1.9 and updated Figure 6-19.</li> <li>■ Added “Handling More Than Three Back-to-Back SETUP Packets.”</li> <li>■ Updated application programming sequence in Section 6.5.2.2.8 and partial power-down description in Section 6.6.1.</li> <li>■ Updated Calculating the Total FIFO Size for DWC_otg</li> <li>■ Added step 13 in Section 6.6.1.4.</li> <li>■ Updated statement about supported interfaces in Section B.1.</li> </ul>
January 2006	2.30a	<ul style="list-style-type: none"> <li>■ hburst, s_hsize notes added in Table 4-2.</li> <li>■ m_hsize changed to 32-bit only in Table 4-3.</li> <li>■ Changed Figure 3-31 and description above it.</li> <li>■ Table 2-1 DWC_otg Configuration Parameters:</li> <li>■ Number of Host Mode Channels description changed</li> <li>■ Total Data FIFO RAM Depth description changed</li> <li>■ Learning queue information added to Section 3.2.3.</li> <li>■ Note added in Section 6.2.3.1.</li> <li>■ Total Data FIFO RAM Depth and Largest Rx Data FIFO Depth descriptions changed in Table 2-1.</li> <li>■ Minimum FIFO Depth Allocation formulas changed. Two bits (20, 22) added to GUSBCFG register in Table 5-9.</li> </ul>

Date	Version	Description
September 2005	2.20a	<ul style="list-style-type: none"> <li>■ Added read and write timing diagrams for AHB Master in Figures 3-13 and 3-14, respectively.</li> <li>■ Added “ULPI Carkit” section and OTG_ULPI_CARKIT configuration parameter.</li> <li>■ Added byte enable and delimiter bit encodings for data FIFOs in Sections 3.3.4.1.1, 3.3.4.1.2, and 3.3.4.1.3, respectively.</li> <li>■ Corrected pseudo-code examples in Interrupt Service Routines for: <ul style="list-style-type: none"> <li>- Interrupt OUT/IN Transactions in Slave Mode</li> <li>- Interrupt OUT/IN Transactions in DMA Mode</li> <li>- Bulk/Control OUT/SETUP and Bulk/Control IN Split Transactions in Slave Mode</li> <li>- Interrupt OUT/IN Split Transactions in Slave Mode</li> <li>- Isochronous OUT/IN Split Transactions in Slave Mode</li> </ul> </li> <li>■ Added the following sections to the Host Programming Model:</li> <li>■ Section 6.3.5, “Selecting the Queue Depth”</li> <li>■ Section 6.3.6, “Handling Babble Conditions”</li> <li>■ Added the following sections to the Device Programming Model:</li> <li>■ Section 6.3.4.3.6, “Handling Babble Conditions”</li> <li>■ Section 6.4.4, “Worst Case Response Time”</li> <li>■ Section 6.4.5, “Choosing the Value of GUSBCFG.USBTrdTim”</li> </ul>
June 2005	2.10a	Added “USB 1.1 Full-Speed Serial Transceiver Interface and I <sup>2</sup> C Interface” section for Full-Speed Serial Transceiver interface and I <sup>2</sup> C support.
February 2005	2.00a	Initial version



# Preface

This databook describes the implementation and use of the Synopsys DesignWare® Cores USB 2.0 Hi-Speed On-The-Go (OTG) controller.

## Product Codes

[Table 4-1](#) lists the products and product codes for the DWC\_otg controller.

**Table 4-1 Product Codes for DWC\_otg Controller**

Component Name or Add-On	Product Code
<b>Base Product</b>	
DWC USB 2.0 HS OTG Controller v4	C435-0
<b>Base Product for Versions 3.30a and Earlier</b>	
DWC USB 2.0 HS OTG Subsys-AHB	3884-0
<b>Add-Ons</b>	
DWC USB HSOTG Hibernate Add-On	6922-0
DWC USB HSOTG HSIC-LPM Add-on	4740-0
<b>End of Life Products/Add-Ons</b>	
DWC USB 2.0 HS OTG-Host Only-AHB	4378-0
DWC USB 2.0 HS OTG-Device Only-AHB	4377-0
DWC USB 2.0 HS OTG-FS Only-AHB	4332-0
DWC USB 2.0 HS OTG-FS Only-ICUSB Add-on	4737-0
DWC USB UTMI-to-UTMI Bridge	B589-0

## Databook Organization

The chapters of this databook are organized as follows:

- [Chapter 1, "Product Overview"](#),
- [Chapter 2, "Architecture"](#)
- [Chapter 3, "Configuration Parameters"](#)
- [Chapter 4, "Signal Interfaces"](#)
- [Chapter 5, "Signal Descriptions"](#)
- [Chapter 6, "Control and Status Registers"](#)
- [Chapter 7, "Register Descriptions"](#)
- [Chapter 8, "Unified Power Format Support"](#)
- [Appendix A, "Example Transfers in Device Scatter/Gather DMA Mode"](#)
- [Appendix B, "Area, Speed, and Power"](#)
- [Appendix C, "DWC\\_otg LPM Add-On Feature"](#)
- [Appendix D, "Active Clock Gating Add-On Feature"](#)
- [Appendix E, "Service Interval-Based Scheduling for ISOC IN Endpoints"](#)
- [Appendix F, "DWC\\_otg HSIC Add-On Feature"](#)
- [Appendix G, "DWC\\_otg IC\\_USB Add-On Feature"](#)
- [Appendix H, "Remote Memory Support for Internal DMA Configurations"](#)
- [Appendix I, "DWC\\_otg Hibernation Add-on Feature"](#)
- [Appendix J, "UTMI-to-UTMI Bridge Add-on Feature"](#)
- [Appendix K, "Chirp\\_on Behavior"](#)
- [Appendix L, "Internal Parameter Descriptions"](#)

## Related Product Documentation

### USB 2.0 HS OTG Controller

Before installing the DWC\_otg, you can download the latest document set, including the Datasheet, Installation Guide, QuickStart, Databook, and Release Notes, at:

[https://www.synopsys.com/dw/ipdir.php?c=dwc\\_usb\\_2\\_0\\_hs\\_otg\\_subsystem-ahb](https://www.synopsys.com/dw/ipdir.php?c=dwc_usb_2_0_hs_otg_subsystem-ahb)

(SolvNet ID required)

### Linux Driver Software for USB 2.0 HS OTG Controller

You can download the Linux Driver Software at:

<https://www.kernel.org/>

Directory: /drivers/usb/dwc2

Synopsys supports these USB drivers.

### Verification IP (VIP)

To run simulations in coreConsultant, you must download and install the Synopsys Verification IP for USB 2.0 HS OTG and AMBA from the SolvNet Download Center, at:

<https://solvnet.synopsys.com/DownloadCenter/dc/product.jsp>

For more information, refer to the *DesignWare Cores USB 2.0 Hi-Speed On-The-Go (OTG) Installation Guide*.

### Synopsys Tools

- *coreConsultant User Guide*, Synopsys, Inc. (included with the coreConsultant tool)  
[http://www.synopsys.com/dw/doc.php/doc/coretools/latest/coreconsultant\\_user.pdf](http://www.synopsys.com/dw/doc.php/doc/coretools/latest/coreconsultant_user.pdf)

## Third-Party and Reference Documentation

### USB Protocol (from USB Implementers Forum)

- *On-The-Go Supplement to the USB 2.0 Specification* (Revision 1.3, December 5, 2006)
- *On-The-Go Supplement to the USB 2.0 Specification* (Revision 2.0, May 8, 2009)
- *Battery Charging Specification* (Revision 1.1, April 15, 2009)
- *High-Speed Inter-Chip USB Electrical Specification* (Version 1.0RC, June 28, 2007)
- *Inter-Chip USB Supplement* (Revision 1.0, March 13, 2006)
- *USB 2.0 Link Power Management Addendum* (Engineering Change Notice to the USB 2.0 specification, July 16, 2007)

### USB Protocol (Other)

- *USB Carkit Specification*, Consumer Electronics Association (CEA 936-A, November 16, 2005)

### PHYs and Transceivers

- *UTMI+ Specification*, Philips Semiconductors (Revision 1.05)  
<http://www.intel.com/technology/usb/spec.htm>
- *UTMI+ Low Pin Interface (ULPI) Specification* (Revision 1.1, October 20, 2004)  
<http://www.ulpi.org>
- *OTG Transceiver Specification* (CEA-2011, Version 4, December 2004)

### SoC Buses

- *AMBA Specification* (Revision 2.0, May 13, 1999)

## Web Resources

- DesignWare IP product information: <http://www.designware.com>
- Your custom DesignWare IP page: <http://www.mydesignware.com>
- Documentation through SolvNetPlus: <https://solvnetplus.synopsys.com> (Synopsys password required)
- Synopsys Common Licensing (SCL): <http://www.synopsys.com/keys>

## Customer Support

Synopsys provides various methods for contacting Customer Support, as follows:

- Prepare the following debug information, if applicable:
  - For environment set-up problems or failures with configuration, simulation, or synthesis that occur within coreConsultant or coreAssembler, select the following menu:

**File > Build Debug Tar-file**

Check all the boxes in the dialog box that apply to your issue. This option gathers all the Synopsys product data needed to begin debugging an issue and writes it to the <core tool startup directory>/debug.tar.gz file.

- ❑ For simulation issues outside of coreConsultant or coreAssembler:
  - Create a waveforms file (such as VPD or VCD).
  - Identify the hierarchy path to the DesignWare instance.
  - Identify the timestamp of any signals or locations in the waveforms that are not understood.
- For the fastest response, enter a case through SolvNetPlus:
  - a. <https://solvnetplus.synopsys.com>



**Note** SolvNetPlus does not support Internet Explorer. Use a supported browser such as Microsoft Edge, Google Chrome, Mozilla Firefox, or Apple Safari.

- b. Click the **Cases** menu and then click **Create a New Case** (below the list of cases).
- c. Complete the mandatory fields that are marked with an asterisk and click **Save**.  
Ensure to include the following:
  - **Product L1:** DesignWare Cores
  - **Product L2:** USB 2.0 OTGAfter creating the case, attach any debug files you created.  
For more information about general usage information, refer to the following article in SolvNetPlus:  
<https://solvnetplus.synopsys.com/s/article/SolvNetPlus-Usage-Help-Resources>
- Or, send an e-mail message to [support\\_center@synopsys.com](mailto:support_center@synopsys.com) (your email will be queued and then, on a first-come, first-served basis, manually routed to the correct support engineer):
  - Include the Product L1 and Product L2 names, and Version number in your e-mail so it can be routed correctly.
  - For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood
  - Attach any debug files you created.
- Or, telephone your local support center:
  - North America:  
Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
  - All other countries:  
<https://www.synopsys.com/support/global-support-centers.html>



# 1

## Product Overview

---

This chapter provides an introduction to the DWC\_otg features, functional blocks, and applications.

This chapter discusses the following topics:

- “System Overview” on page [37](#)
- “Features List” on page [39](#)
- “Changes from the Previous Release” on page [46](#)
- “Known Issues and Limitations” on page [47](#)
- “Compliance With Standards” on page [55](#)
- “Certification” on page [56](#)
- “Coding and Design Guidelines and Exceptions” on page [56](#)
- “Speed and Clock Requirements” on page [57](#)
- “Power and Area” on page [58](#)

**⚠ Attention**

From the 4.20a release onwards, the following Controller Configuration Options are not supported.

If support is required for your specific project, contact Synopsys IP Support.

Since many of these features are documented along with other aspects of the Controller at many places. Removing information about these features may make this documentation more complicated and tougher to use. So, the documents packaged with the controller will continue to have information about these features.

- **Mode of Operation (OTG\_MODE):** The following values are not supported:

- HNP- and SRP-Capable OTG (Device & Host)
- SRP-Capable OTG (Device & Host)
- SRP-Capable OTG Device
- SRP-Capable OTG Host

Note: USB-IF has announced End of Life for USB OTG (Support for HNP and SRP)

[USB-IF Notification](#)

- **Architecture (OTG\_ARCHITECTURE):** External DMA (is not supported)

- **Enable Dedicated Transmit FIFO for device IN Endpoints (OTG\_EN\_DED\_TX\_FIFO)**

- No (is not supported)
- This means that only Dedicated TX FIFO Mode Configuration is supported and this is recommended to provide flexibility for applications like Video/Audio streaming which are becoming more prevalent

- **Enable Dynamic FIFO Sizing? (OTG\_DFIFO\_DYNAMIC)**

- No (is not supported)
- This means that Dynamic FIFO Sizing must always be enabled and this is recommended to provide flexibility for applications like Video/Audio streaming which are becoming more prevalent.

- **Enable Power Optimization? (OTG\_EN\_PWROPT)**

- ExtendedHibernation (is not supported)

- **Enable UPF Power Clamps? (OTG\_PWR\_CLAMP)**

- No (is not supported)
- The power clamps required at power domain boundaries of the Controller as specified in the UPF file packaged with the controller

- **Exceptional Control Transfer Flow Support? (OTG\_EXCP\_CNTL\_XFER\_FLOW)**

- No (is not supported)
- The controller in Device mode of operation will always support the handling of Exceptional Control Transfer scenarios

- **Enable UTMI-To-UTMI Bridge Component? (DWC\_U2UB\_EN)**

- Enable (is not supported)
- This Add-on feature is not support with this release of the controller

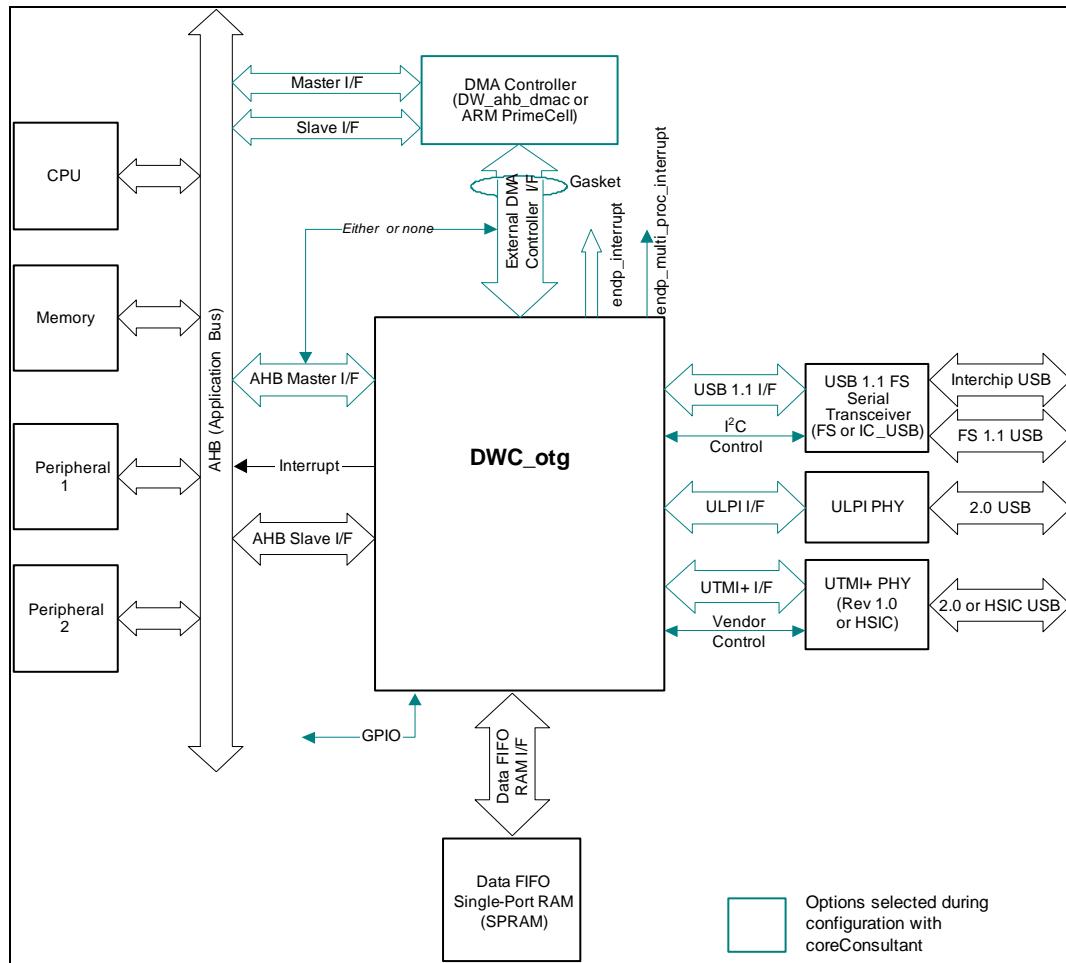
## 1.1 System Overview

This section contains a high-level description of DWC\_otg interfaces and functional blocks.

### 1.1.1 System-Level Block Diagram

[Figure 1-1](#) shows the DWC\_otg controller in a typical system. The controller interfaces are summarized in the following subsections.

**Figure 1-1 System-Level Block Diagram**



## 1.1.2 Interfaces

This topic contains descriptions for all the major interfaces in DWC\_otg.

### 1.1.2.1 AHB Slave and Data RAM

These interfaces are present in all configurations of the DWC\_otg controller:

- AHB Slave, which provides the micro-controller with read and write access to the controller's Control and Status Registers (CSRs), Data FIFO, and queues
- Data RAM interface, which connects to an external Single-Port FIFO RAM (SPRAM) for transaction data storage. Configurable, 32 to 32,768 deep by 35 bits wide (32 data bits plus 3 control bits)

### 1.1.2.2 USB 2.0 PHYs and USB 1.1 Full-Speed Serial Transceiver

The DWC\_otg controller supports any or all of these PHY and serial transceiver interfaces. One or more must be selected during configuration with the coreConsultant tool.

- UTMI+ Level 3 PHY interface (Revision 1.0 or HSIC).

The UTMI+ PHY can be unidirectional or bidirectional, with 8-, 16-, or 8/16-bit data buses (software selectable).

The UTMI+ interface can be configured to work with Revision 1.0 compliant PHY or HSIC compliant PHY

- ULPI PHY interface (Revision 1.1)

The ULPI PHY can be unidirectional or bidirectional, with 8-bit SDR or 4-bit DDR buses (software selectable).

- USB 1.1 Full-Speed Serial Transceiver interface.

The USB 1.1 Full-Speed Serial Transceiver can have a dedicated interface or share pins on UTMI+ or ULPI interfaces for off-chip PHYs (software-selectable).

The Full-Speed Serial Transceiver interface may be configured as either a USB 1.1 6-pin/3-pin interface or an IC\_USB interface when you choose the IC\_USB option.

The IC\_USB interface is an additional dedicated interface. Pin sharing with UTMI+ or ULPI is not available for this interface. The IC\_USB interface, once selected is available irrespective of FS pin shared option. The IC\_USB interface is an add-on feature that requires an additional DWC-HSOTG-FS-ICUSB license.



**Note** The DWC\_otg controller can support two ports in parallel (for example, UTMI+ in parallel with the dedicated FS 1.1). However, both ports cannot operate simultaneously. At any given time, only one port is active. While switching from one port to another port, the application must provide a software reset.

### 1.1.2.3 Optional Interfaces

These interfaces are optional and are built only when appropriate parameters are selected during configuration with the coreConsultant tool:

- Internal DMA controller (AHB Master) enables the controller to act as a Master on the AHB to transfer data to and from the AHB.
- Vendor Control interfaces to ULPI and UTMI+ PHY registers
- I<sup>2</sup>C interface for OTG control in USB 1.1 FS mode
- General Purpose I/O (GPIO) interface
- SOF update toggle port (not used for DWC\_otg)
- SOF input count port (not used for DWC\_otg)
- Descriptor-Based Scatter/Gather DMA controller for Device and Host mode. (Descriptor-Based Congruent-Sequential DMA is not supported.)
- The controller supports the Scatter Gather DMA operation in both Device and Host mode. Select this option if the Device mode has Isochronous endpoint support to gain more performance. Note that hubs (split transfers) are not supported in Host Scatter Gather DMA mode of operation. Split transfers are supported only in Host Buffer DMA (Internal DMA) mode of operation.
- Multi Processor Interrupt for Device Mode.

This feature supports endpoint-specific interrupt mask registers and endpoint-specific interrupt signals from the controller. This feature can be used in a multi-processor environment in which each endpoint can be controlled by a different processor.

For more information on DWC\_otg interfaces, see “[Architecture](#)” on page 59.

### 1.1.3 Transmit and Receive FIFOs

A transmit and receive FIFO interface on the DWC\_otg controller are used to move data in and out of the Data FIFO RAM. These are classified as periodic (for isochronous and interrupt transfers) or non-periodic (for bulk and control transfers) and are summarized as follows

A single Receive FIFO for all host IN and device OUT transfers

- Single Common Non-Periodic Tx FIFO for Non-Periodic Host OUT transfers.
- Optional single Periodic Transmit FIFO for periodic host OUT transfers.
- Optional Dedicated Tx FIFOs for each IN endpoint for periodic / non-periodic Device IN transfers.
- Host mode interrupt OUT and Device mode interrupt IN (in Shared FIFO operation only) transfers can be programmed to go through either the Non-periodic Transmit FIFO or Periodic Transmit FIFOs.<sup>1</sup>

For more information, see “[Architecture](#)” on page 59.

## 1.2 Features List

The DWC\_otg includes the following features, listed by category.

1. In Shared FIFO operation, for mass storage devices with Control-Bulk-Interrupt (CBI) protocol, the interrupt IN endpoint does not use a periodic endpoint (normally used for status updates at the end of bulk data transfers). When a single mass storage device is connected, and software can predict when IN interrupts are received, the IN interrupt can be sent through the Non-periodic Transmit FIFO controller, saving a periodic FIFO and the associated memory for it.

### 1.2.1 General Features

- <sup>1</sup>Software configurable to OTG1.3 and OTG2.0 modes of operation
  - OTG2.0 Supports ADP (Attach detection Protocol)
- Support for the following speeds:
  - High-Speed (HS, 480-Mbps),
  - Full-Speed (FS, 12-Mbps) and
  - Low-Speed (LS, 1.5-Mbps) modes
- Multiple options available for low power operations
- Multiple DMA/non DMA mode access support on the application side
- Multiple Interface support on the MAC-Phy
- Supports different clocks for AHB and the PHY interfaces for ease of integration
- Supports up to 16 bidirectional endpoints, including control endpoint 0.
- Low speed is not supported for DWC\_otg as a device with a UTMI+ PHY.
- <sup>2</sup>Supports Session Request Protocol (SRP)
- <sup>3</sup>Supports Host Negotiation Protocol (HNP)
- Supports up to 16 host channels. In Host mode, when the number of device endpoints to be supported is more than the number of host channels, software can reprogram the channels to support up to 127 devices, each having 32 endpoints (IN + OUT), for a maximum of 4,064 endpoints.
- Supports the external hub connection in Host Buffer DMA mode, Slave mode, and External DMA mode.

**Note:** DWC\_otg controller in Host Scatter Gather DMA mode of operation does not support split transfers in the hardware. Only Buffer DMA mode of operation supports split transfers in the hardware and hence generic root hub.

- Includes automatic ping capabilities
- Supports the Keep-Alive in Low-Speed mode and SOFs in High/Full-Speed modes

### 1.2.2 Configurable Features

- Uses the coreConsultant utility to configure the controller to user requirements
- Ability to choose multiple power rails for low power modes
- Choice of multiple DMA modes of operation
- Choice of type of Mac-Phy interface required

### 1.2.3 Application Interface Features

- Interfaces for the application via the AHB:
    - AHB Slave interface for accessing Control and Status Registers (CSRs), the Data FIFO, and queues
1. Feature is no longer supported.
  2. Feature is no longer supported.
  3. Feature is no longer supported.

- Optional AHB Master interface for Data FIFO access when Internal DMA is enabled
- Supports AHB clock frequencies up to 270 MHz with suitable technology (tested with a standard 65 nm technology with SDF, and without post-layout delays, clock tree synthesis or gate-level simulations) for certain configurations only.
- Supports only 32-bit data on the AHB.
- Supports Little or Big Endian mode (selectable by pin).
- Supports INCR4, INCR8, INCR16, INCR, and SINGLE transfers on the AHB Slave interface.
- Supports Split, Retry, and Error AHB responses on the AHB Master interface. Split and retry responses are not generated on the AHB Slave interface.
- Software-selectable AHB burst type on AHB Master interface in DMA mode
  - If INCR4 is chosen, the controller only uses INCR/INCR4, or Single.
  - If INCR8 is chosen, the controller normally uses INCR8, but at the beginning and at the end of a transfer, it can use INCR or Single, depending on the size of the transfer.
  - If INCR16 is chosen, controller normally uses INCR16, but at the beginning and at the end of a transfer, it can use INCR or Single, depending on the size of the transfer.
- Handles the fixed burst address alignment. For example, INCR16 is used only when lower addresses [5:0] are all 0.
- Generates AHB Busy cycles on the AHB Master interface
- Takes care of the 1KB boundary breakup

### 1.2.4 MAC-PHY Interface Features

- Support for the following MAC-Phy Interfaces:
  - UTMI 8/16,
  - ULPI,
  - HSIC,
  - IC\_USB (for Low/Full speed),
  - FS Shared on UTMI
- Supports the UTMI+ Level 3 interface (Revision 1.0, February 25th, 2004). 8-, 16-, and 8/16-bit data buses are supported.
- Supports ULPI interface (Revision 1.1rc, September 1st, 2004), 8-bit SDR, 4-bit DDR, 6-pin Serial, 3-pin Serial and Carkit.
- The UTMI+ L3 and ULPI can both exist and be selected by software, or only the required interface can be specified during coreConsultant configuration.
- HSIC interface can be selected only if UTMI+ is chosen. HSIC interface cannot be selected otherwise.



**Note** The HSIC feature is an add-on feature that requires an additional DWC-HSOTG-HSIC license.

---

- I<sup>2</sup>C interface (for support of *Mini USB Analog Carkit Interface Specification*, CEA-936, Revision 2). Not intended for use with other devices.
- Supports UTMI-to-UTMI Bridge (See Appendix J, “[UTMI-to-UTMI Bridge Add-on Feature](#)”)

### 1.2.5 System Memory Architecture

- Supports Slave, <sup>1</sup>External DMA Controller Interface, or Internal DMA modes
- Optional Descriptor-Based Scatter/Gather DMA operation when Internal DMA mode is chosen
- Includes optional interface to an external DMA controller; data is transferred through the AHB Slave interface.

1. No longer supported.

### 1.2.6 Non-DWORD Alignment Support

- Host Mode:
  - Scatter Gather DMA mode, IN and OUT transfers - Non-DWORD alignment of buffer addresses is supported
  - Buffer DMA and Slave mode, IN and OUT transfers - Non-DWORD alignment of buffer addresses is not supported
- Device Mode:
  - Scatter Gather DMA mode, IN and OUT transfers - Non-DWORD alignment of buffer addresses is supported
  - Buffer DMA and Slave mode, IN and OUT transfers - Non-DWORD alignment of buffer addresses is not supported



**Note** Non-DWORD alignment support is available only for buffer addresses and not for descriptors.

### 1.2.7 Internal Memory Features

- Optional support for a dedicated transmit FIFO for each of the device IN endpoints in Slave and DMA modes. Each FIFO can hold multiple packets.
- Includes an optional interface for Remote Memory Support used to signal the controller of a DMA write complete event on the system.
- <sup>1</sup>Supports packet-based, Dynamic FIFO memory allocation for endpoints for small FIFOs and flexible, efficient use of RAM.
- Uses single-port RAM instead of dual-port RAM for smaller area and lower power.
- Provides support to change an endpoint's FIFO memory size during transfers.
- Supports endpoint FIFO sizes that are not powers of 2, to allow the use of contiguous memory locations.
- Shares the hardware registers in the Host and Device modes to reduce gate count.

1. Feature is no longer supported

- Optional support for Transmit and Receive thresholding in DMA mode when dedicated Tx FIFO is selected in Device mode. Thresholding and threshold length selectable through global registers. For supporting thresholding, the AHB must be run at 60 MHz or higher.



- *Dedicated FIFO operation* refers to the controller configuration in which each Device mode IN endpoint has individual transmit FIFOs.
- <sup>a</sup>*Shared FIFO operation* refers to the controller configuration in which all non-periodic IN endpoints share a common TX FIFO and periodic IN endpoints have separate individual FIFOs. In Shared FIFO operation, for mass storage devices with Control-Bulk-Interrupt (CBI) protocol, the interrupt IN endpoint does not use a periodic endpoint (normally used for status updates at the end of bulk data transfers). When a single mass storage device is connected, and software can predict when IN interrupts are received, the IN interrupt can be sent through the non-periodic Transmit FIFO controller, saving a periodic FIFO and the associated memory for it.

- a. No longer supported.

## 1.2.8 Software Features

To increase flexibility and reduce gate count, certain functions are implemented in software:

- Software assists hardware for Device mode non-periodic IN sequencing (applicable only in Shared FIFO operation).
- Software handles USB commands (SETUP transactions are detected and their command payloads are forwarded to the application for decoding).
- Software handles USB errors.

## 1.2.9 Power Optimization Features

- Active Clock Gating Feature:

Allows the controller to reduce dynamic power consumption by gating the AHB and PHY clocks internally within the modules of the controller during the IDLE periods between the USB and AHB traffic. It also gates the RAM clock output when the SPRAM is not accessed by the controller.

- Link Power Management (LPM) Support



This is an add-on feature that requires an additional DWC-HSOTG-LPM license.

Additional flexibility for handling L1 requests during Audio Traffic in use cases that target USB Audio Class 3.0

- Several power-saving features including two power rails for advanced power management. You can choose any of the following power saving options according to your requirements:
  - Clock gating
  - Partial Power Down

- Hibernation - In hibernation (enabled when OTG\_EN\_PWRLOPT=2 or 3), the DWC\_otg\_core module that comprises approximately 95% of the gate count can be power gated in Host and Device mode during Suspend. DWC\_otg\_pmu is capable of detecting Resume, Remote Wakeup, SRP, or Connect during Suspend and wake up the application when the controller is in hibernation state.
- <sup>1</sup>Extended Hibernation - In extended hibernation (enabled when OTG\_EN\_PWRLOPT=3), the DWC\_otg\_wpc\_slv, DWC\_otg\_power\_dn, and DWC\_otg\_pmu modules are power gated. This feature is supported in Device mode only. The DWC\_otg\_piup module is capable of detecting Resume and can respond to IN/OUT tokens with NAK handshake and store one Setup packet till the application is powered up. This feature is useful in cases where the complete SoC and the CPU is power gated and part of the USB module is powered on to detect activity from the Host.



**Note** Hibernation and Extended Hibernation features are add-on features that require an additional DWC-HSOTG-HIBERNATION license

- PHY clock gating support during USB Suspend, LPM, and Session-Off modes
- AHB clock gating support during USB Suspend and Session-Off modes
- Partial power-off during USB Suspend and Session-Off modes
- Hierarchy to support multiple power rails to enable Hibernation feature during suspend
- Input signals to powered-off blocks driven to safe 0
- Data FIFO RAM chip-select de-asserted when not active
- Data FIFO RAM clock-gating support
- Switching to lower frequency 32-KHz clock support for both Device and Host modes during USB Suspend, LPM, and Session-Off modes

### 1.2.10 Differences Between OTG Revision 1.3 and OTG Revision 2.0

Table 1-1 lists the differences between the OTG Revision 1.3 and OTG Revision 2.0 features.

**Table 1-1 Differences Between OTG Revision 1.3 and Revision 2.0**

OTG Revision 1.3	OTG Revision 2.0
Attach Detection Protocol (ADP) is not supported	Attach Detection Protocol (ADP) is supported
Accessory Charger Adapter (ACA) is not supported	Accessory Charger Adapter (ACA) is supported (only if BC Support =1 is configured through coreConsultant). For more information on ACA, see <i>Battery Charging Specification</i> (Revision 1.1).
Supports both data line and VBUS pulsing method of SRP.	The VBUS pulsing method of SRP is no longer supported. Only data line pulsing is supported.

1. No longer supported

## 1.3 Changes from the Previous Release

This release of the DWC\_otg core contains the following changes:

VTB enhanced to use SVT VIPs instead of VERA-VMT VIPs.

For details see, Chapter 4 of the *DesignWare Cores USB 2.0 Hi-Speed On-The-Go (OTG) User Guide*.



From the v4.20a release onwards, a few Controller Configuration options are no longer supported. For more details, see [Note](#) on page [36](#).

---

## 1.4 Known Issues and Limitations

This section covers the following topics:

- “Feature and Verification Limitations” on page 48
- “Worst-Case Inter-Packet Gap and Maximum Number of Cascaded Hubs Supported by Device Controller” on page 49
- “Full-Scale Simulation” on page 50
- “Thresholding” on page 50
- “Synthesis” on page 50
- “coreConsultant Configuration Parameters” on page 51
- “DRC Violations” on page 51
- “System-Level Testing” on page 52
- “Software” on page 53
- “Hardware Testing (Slave and Internal DMA)” on page 53
- “MemoryMap” on page 53
- “Limitations in the VTB” on page 54
- “Spyglass” on page 54

### 1.4.1 Feature and Verification Limitations

- The following features have not been tested on hardware, but have been tested in simulations:
  - ULPI-DDR
  - USB 1.1 FS Carkit
  - ULPI Carkit
- The Synopsys Reference driver has not been tested in USB1.1 Low Speed Transceiver mode. This feature has been tested in simulations.
- When the DWC\_U2UB Add-on feature is enabled, no new U2UB specific tests are present in the Setup and Run Simulations Activity. However, U2UB-specific tests are present in the vtb and the details are provided in the *DesignWare Cores USB 2.0 Hi-Speed On-The-Go (OTG) User Guide*.
- The U2UB-specific tests in the vtb can be run only with VCS. They cannot be run with NC Verilog or with MTI Simulators.
- When the DWC\_U2UB controller is exiting from Suspend, the u2u\_dev\_utmi\_clk or u2u\_host\_utmi\_clk are driven by aux\_clk for 2 cycles initially after which the u2u\_dev\_utmi\_clk or u2u\_host\_utmi\_clk switch to the respective utmi\_host\_clk or utmi\_dev\_clk.

This should not lead to functional issues as the Resume and Remote Wakeup signaling lasts around 1 or more milliseconds.

- When using the VCS NLP flow to run simulation tests in the VTB, you may see the following error message. This error message can be ignored because it does not cause any functional impact.

UPF error messages:

```
[0 fs] [ERROR] [LP_PSW_CTRL_INIT_INVALID] Signal 'otg_vtb_top/pwr_switch' connected to control port 'sw_control' of power switch 'otg_vtb_top/power_switch' started with an invalid value 'StX'.
```

```
[0 fs] [INFO] [LP_PSW_INIT_STATE] Power switch 'otg_vtb_top/power_switch' started in INVALID (ERROR) state.
```

```
[10000 fs] [INFO] [LP_PSW_STATE_CHANGE] State of power switch 'otg_vtb_top/power_switch' changed from INVALID (ERROR) to ON (ON)
```

- Host HNP (run\_vcs\_nlp hst\_int\_dma\_sof\_utmi\_fs\_spd hst\_int\_dma\_sof\_utmi\_fs\_spd VTB\_OTG\_HOST VTB\_DMA\_TESTS VTB\_SOF\_TESTS USE\_UTMI\_PHY USBTE\_FS\_SPD) and Device Extended Hibernation (run\_vcs\_nlp dev\_int\_dma\_utmi\_xhib dev\_int\_dma\_utmi\_xhib VTB\_OTG\_DEV VTB\_DMA\_TESTS USE\_UTMI\_PHY VTB\_DEV\_XHIB) Test - may fail when run with VCS NLP. These tests will be removed in the next release.

## 1.4.2 Worst-Case Inter-Packet Gap and Maximum Number of Cascaded Hubs Supported by Device Controller

When the DWC\_otg controller acts as a device, there is a worst-case response time for a receive followed by a receive, as per the *UTMI Specification*. This worst-case response time depends on the AHB clock frequency.

The controller registers are in the AHB domain, and the controller does not accept another token before updating these register values. This worst-case value is seven PHY clocks when the AHB clock is the same as the PHY clock. When AHB clock is faster, this value is smaller.

The following scenarios are for a receive followed by a receive:

- Any response token from host (ACK/NAK/NYET/STALL) followed by a token
- As a special case, ISOC OUT Data packet followed by the next token

If the worst-case condition occurs,

- For non-isochronous tokens (including SETUP token), the controller responds with a NAK or may drop the tokens without responding. The host interprets this as a timeout condition and retries.
- For isochronous transfers, the controller sets the incomplISOCIN and incomplISOCOUT interrupts to inform the application that isochronous IN/OUT packets were dropped.

At the system level, this limitation can reduce the performance of the device controller because the controller drops or sends NAK to the token which is sent by the host too close to end of the preceding token. For bulk, interrupt, and control transactions, the system recovers as the host retires the token, and the transactions will eventually complete. For isochronous transactions, the host proceeds to the next transaction scheduled for the next bInterval.

A significant performance degradation occurs in the following cases:

- The device is connected to a host which is capable of consistently sending a token within 88-96 bit times, that is, six to seven PHY clocks from the end of the preceding ISOC OUT token.
- The device is connected to a host through multiple layers of hubs such that the inter-packet gap between an ISOC OUT token and the following token shrinks to less than seven PHY clocks.

However, in neither of these cases, the system stops responding nor enters an unrecoverable state.

From 4.10a release, the controller can handle the minimum inter-packet gap as per the *UTMI Specification* in the previously-mentioned isochronous scenario (ISOC OUT Data Packet followed by the next token). In practice, it has been observed that the USB Host implementation are unable to send tokens with the worst-case inter-packet gap of 88-96 bit times. Refer to the “Impact of System Clocks on Minimum Inter-Packet Gap and Number of Cascaded Hubs Supported by the Device” and “Speed and Clock Requirements” sections in the databook for more details on the impact of the AHB clock and PHY clock frequencies on the following characteristics of the device controller:

- The worst-case inter-packet gap that the device controller can support
- The number of cascaded hubs after which the device can be connected

### 1.4.3 Full-Scale Simulation

Scale-down mode is used for simulation; do not use non-scale-down mode for simulation. Full-scale verification is performed as part of hardware validation in FPGA.

### 1.4.4 Thresholding

- When enabling thresholding, the AHB must run at 60 MHz or faster.
- Synopsys recommends that you do not enable DTHRCTL.RxThrEn, because it may cause issues in the RxFIFO especially during error conditions such as RxError and Babble.

### 1.4.5 Synthesis

- Presto (HDL Compiler) should always be enabled because Verilog 2000 syntax is used in RTL.
- The following warnings are displayed in the check\_test report when synthesizing with scan. Warnings (1) and (2) are due to the gated clocks. Warning (3) is due to the DDR data path for ULPI, where the clock is used as the select for the data MUX.

(1) Warning: Data pin **\*\*inside\*\*** of cell `hclk_gated (**out_port**)` is driven by a clock/enable signal. (TEST-131) Information: The network contains: `hclk_gated`, `hclk`, `U_DWC_otg_clkrst/U_DWC_otg_mux_5/U6/Y`, `U_DWC_otg_clkrst/U_DWC_otg_mux_5/U6/B1`, `U_DWC_otg_clkrst/U_DWC_otg_mux_5/U6/A0`. (TEST-281)

(2) Warning: Data pin **\*\*inside\*\*** of cell `phy_clk (**out_port**)` is driven by a clock/enable signal. (TEST-131) Information: The network contains: `phy_clk`, `utmi_clk`, `U_DWC_otg_clkrst/U_DWC_otg_mux_6/U4/Y`, `U_DWC_otg_clkrst/U_DWC_otg_mux_6/U4/B1`, `U_DWC_otg_clkrst/U45/B0`, `U_DWC_otg_clkrst/U45/Y`, `U_DWC_otg_clkrst/U49/B0`. (TEST-281) Information: There is one other pin with the same violation. (TEST-172)

(3) Warning: Data pin Y of cell `U_DWC_otg_piu/U_DWC_otg_piu_ulpi/U365` (NAND3X6) is driven by a clock/enable signal. (TEST-131) Information: The network contains: `ulpi_clk`, `U_DWC_otg_piu/U_DWC_otg_piu_ulpi/U365/C`, `U_DWC_otg_piu/U_DWC_otg_pi_u_ulpi/U397/B`, `U_DWC_otg_pi_u/U_DWC_otg_pi_u_ulpi/U400/C1`, `U_DWC_otg_pi_u/U_DWC_otg_pi_u_ulpi/U403/Y`, `U_DWC_otg_pi_u/U_DWC_otg_pi_u_ulpi/U403/AN`, `U_DWC_otg_pi_u/U_DWC_otg_pi_u_ulpi/U456/A`, `U_DWC_otg_pi_u/U_DWC_otg_pi_u_ulpi/U748/A1`, `U_DWC_otg_pi_u/U_DWC_otg_pi_u_ulpi/U750/A1`, `U_DWC_otg_pi_u/U_DWC_otg_pi_u_ulpi/U755/Y`, `U_DWC_otg_pi_u/U_DWC_otg_pi_u_ulpi/U755/AN`, `U_DWC_otg_clkrst/U48/A0`. (TEST-281)

- Synopsys recommends the One-Pass compile strategy available through coreConsultant for better quality of results. This strategy is part of the Design Compiler reference synthesis flow.
- When synthesis is run through coreConsultant and if UPF is enabled, due a limitation, ATPG cannot be run.
- When synthesis is run with DC-Opto Strategy, it observed that some paths are reported as having timing violations but these are not valid violations and will not be observed if another DC strategy is chosen. The Synopsys team is investigating if this is a tool limitation.
- To improve the fault coverage of design to 99%, it is recommended to follow the two-run approach. When TetraMax is run on scan-inserted netlist, it runs in Internal/Uncompressed mode by default. Coverage of the faults in the design can be improved by running the TetraMax tool first in the Compression mode, then followed by Internal mode. This two-run approach covers the ATPG untestable faults in decompressor and compressor instances resulting in improved coverage numbers.

To implement two-run approach to improve coverage, perform the following steps:

- a. Run **Compression Mode ATPG**. Write-out a fault list of the detected faults.
- b. Run **Internal Mode DRC**. Add Faults. Read Detected faults from Step a. Run **ATPG** to target the remaining undetected faults.
- In configurations with IC\_USB Full-Speed Serial Transceiver interface enabled (OTG\_ENABLE\_IC\_USB=1), Design Compiler may report setup timing violations from,  
 Startpoint: `ic_usb_iddig` to Endpoint: `utmifs_dp_rpul_en_n`  
 Startpoint: `ic_usb_iddig` to Endpoint: `utmifs_fs_edge_sel`  
 These timing violations are observed with Design Compiler and can be ignored as these are a result of a tool limitation.

## 1.4.6 coreConsultant Configuration Parameters

### Power Optimization

If you select “Partial Power Down”, “Hibernation”, or “Extended Hibernation” for the “Enable Power Optimization” (OTG\_EN\_PWROPT) option during configuration, the coreConsultant test clock in the Specify Clock(s) section selects all clocks as scan clock. You must manually deselect all clocks and select only utmi\_clk as the test clock.

### Fix Hold

In cC GUI -> Specify clock section, if the Fix Hold option is checked, then Design Compiler attempts to fix hold violations for the flip-flops in the scan chain. This may lead to unnecessary long DC run times. With the Fix hold option unchecked, Design Compiler run times should be significantly quicker.

### Endpoint Direction

Synopsys recommends that you configure Direction of Endpoint  $n$  to be a bi-directional endpoint (OTG\_EP\_DIR\_n = 0)

## 1.4.7 DRC Violations

During synthesis with scan insertion, Design Compiler may report DRC violations prior to design-for-test due to the reset signal used in the combinational logic that feeds the flip-flop’s D input.

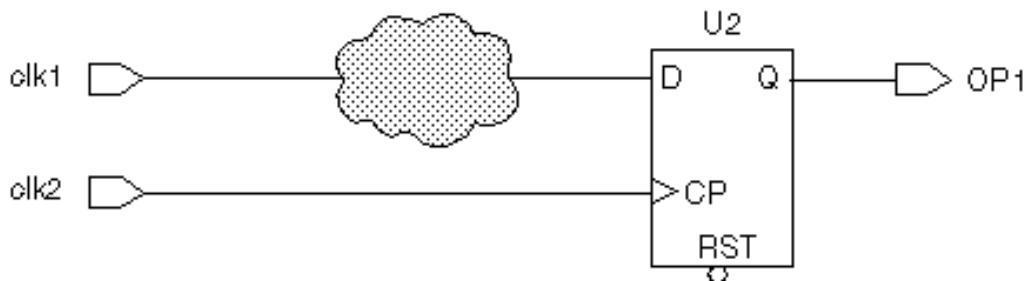
### Example 1-1 U\_DWC\_otg\_always\_on, U\_DWC\_otg\_pi, U\_DWC\_otg\_piulpi, piu\_termselect\_r

The following code pertains to the DWC\_otg\_wpc.v module:

```
assign wpc_termselect      = !wpc_rst_n ? HS_TERM :  
    wpc_state[IDLE] ? mac_termselect_cl :  
    (iddig_f && !bvalid_c) ? HS_TERM :  
    ((wpc_state[RESUME] & sh2pd_susp_to_reset) ?  
     HS_TERM : FS_TERM);
```

For more information on this violation, see the “Clocks that Interact With Register Input” section of the *DFTMAX Design-for-Test User Guide* at: <https://solvnet.synopsys.com>, specifically, Figure 2-10, which is reprinted here:

**Figure 1-2 Clock Interacting with Register Input (Figure Reprinted from the DFT User Guide)**



clk1 is the reset signal (wpc\_prst\_n) and clk2 is ulpi\_clk.

This issue can be waived, because during normal functional mode, the asynchronous reset signal to the flip-flop (which is also clk1) resets the flop, and thus presents no timing hazard. In Scan mode, Design Compiler infers a scan MUX after the combinational logic and before the flip-flop's D input; hence, the combinational cloud path is not selected during the scan.



DRC violations are reported for some constraints like the following:

- min\_delay and max\_transition
- Constraints that are defined in the technology library that is being used.

Often, the technology libraries are provided with highly restrictive constraints to get the best out of the tool. This results in violations reported for values which are still within the accepted range. In general, these would be taken care in the subsequent stages of the tapeout cycle. These are not valid design issues, and can be ignored.

#### 1.4.8 System-Level Testing

You must use the DesignWare Verification IP from the Synopsys DesignWare general release for your system-level testbench, available at:

<https://solvnet.synopsys.com/DownloadCenter/dc/product.jsp>

See the DesignWare Verification IP documents at:

<http://www.synopsys.com/verification/verification-ip.html>

## 1.4.9 Software

For version 4.20a, software will be available shortly. Please contact sales for additional information.

### Linux Driver Software for USB 2.0 HS OTG Controller

You can download the Linux Driver Software at:

<https://www.kernel.org/>

Directory: /drivers/usb/dwc2

Synopsys supports these USB drivers.

## 1.4.10 Hardware Testing (Slave and Internal DMA)

- ULPI-DDR testing has not been performed on an FPGA platform. This feature has been tested in simulations.
- ULPI Carkit testing has not been performed on hardware. This feature has been tested in simulations
- The following features have not been tested on hardware, but have been tested in simulations:
  - ULPI-DDR
  - USB 1.1 FS Carkit
  - ULPI Carkit
- The Synopsys Reference driver has not been tested in USB1.1 Low Speed Transceiver mode. This feature has been tested in simulations.

## 1.4.11 MemoryMap

In the DWC\_otg core, the GNPTXFSIZ register has different behavior and bit description in the following modes:

- Host Mode and Device Shared FIFO Mode
- Device Dedicated FIFO Mode

The present version of the IP-XACT is unable to capture the dynamic behavior of this register based on the Host/Device mode of operation. Therefore, when the core is configured, the log file contains the following error message during the xml\_verify stage in the coreConsultant:

```
SCR 7.2 violated for component with VLVN: Synopsys DesignWareCores DWC_otg 4.20a
The register fields NPTxFDep and INEPTxF0Dep in register GNPTXFSIZ have overlapping
ranges.
*** Checking summary ***
*** <syntax errors> <semantic errors> : <VLVN> | <file name> ***
 0 1 : component Synopsys DesignWareCores DWC_otg 4.20a
Total errors: 1
*** SCR violation summary (SCR number, total violations) ***
7.2 1
```

You can ignore this error message. It does not cause any functional issue.

### 1.4.12 Limitations in the VTB

- VTB needs minimum two endpoints in addition to control endpoint for all the tests to run. Testcases for single endpoint configurations fail while trying to initiate transfers on non-existent endpoints.
- In configurations with Dynamic FIFO Sizing enabled, when DFIFO size is less, you must change the parameters defining MPS and Packet Count in <install\_dir>/sim/SoC\_sim/vtb/test\_common.h to a suitable value.

If the value of MPS and Packet Count is not correct, the test environment displays the following error messages:

VTB FIFO ERROR:OTG\_DFIFO\_DEPTH is too small to meet your usage  
and

RXFIFO ERROR:Your RX FIFO CONFIG is NOT Enough in the cC TO %d WORDS, You'd better re-calculate the size and configure the core again

### 1.4.13 Spyglass

- DFT goals, when enabled with the other goals (Lint, CDC, RDC), might result in false RDC violations, due to Spyglass tool issue.
- Hence, it is recommended that while running Spyglass from the coreConsultant GUI, DFT goals (dft\_abstract and dft\_best\_practice) should be run separately and must not be enabled with other goals.



**Note** VC Spyglass SGUM option in the Run Spyglass activity is not supported. Do not select this option.

---

## 1.5 Compliance With Standards

The DWC\_otg controller complies with these standards:

- *On-The-Go Supplement to the USB 2.0 Specification* (Revision 1.3)
- Complies with the *On-The-Go Supplement to the USB 2.0 Specification* (Revision 2.0). For more information on the feature differences between Revision 1.3 and Revision 2.0, see [Table 1-1](#).
- Supports *Battery Charging Specification* (Revision 1.1)
- Complies with the *USB 2.0 Link Power Management Addendum* applied to *Universal Serial Bus Specification* (Revision 2.0)
- *UTMI+ Specification*, Phillips Semiconductor (Revision 1.0, February 25th, 2004)
- *UTMI+ Low Pin Interface (ULPI) Specification* (Revision 1.1, October 20, 2004).
- *OTG Transceiver Specification* (CEA-2011, Version 4, December 2004)
- *AMBA Specification* (Revision 2.0, May 13, 1999)
- *USB 2.0 Link Power Management Addendum* (Engineering Change Notice to the USB 2.0 specification, July 16, 2007)
- *High-Speed Inter-Chip USB Electrical Specification* (Version 1.0RC, June 28, 2007)
- *Inter-Chip USB Supplement* (Revision 1.0, March 13, 2006)

## 1.6 Certification

In UTMI mode, the DWC Hi-Speed USB On-The-Go Controller (DWC USB 2.0 HS OTG-AHB) has received the Hi-Speed USB OTG certification with the Synopsys UTMI+ PHYs as follows:

- DWC USB2 PHY OTG, TSMC 90G
- DWC USB2 PHY-TSMC.13G, OTG (TSMC)

In ULPI mode, the DWC Hi-Speed USB On-The-Go Controller (DWC USB 2.0 HS OTG-AHB) has received the Hi-Speed USB OTG certification as follows:

- DWC Hi-Speed USB On-The-Go with Philips ISP 1504 ULPI PHY
- DWC Hi-Speed USB On-The-Go with SMSC USB 3300 ULPI PHY
- DWC Hi-Speed USB On-The-Go with Philips ISP 1508 ULPI PHY

In USB 1.1 FS Transceiver mode, the DWC Hi-Speed USB On-The-Go Controller (DWC USB 2.0 HS OTG-AHB) has received the Full-Speed USB OTG certification as follows:

- DWC Hi-Speed USB On-The-Go with Philips ISP 1301 FS PHY
- DWC Hi-Speed USB On-The-Go with Philips ISP 1302 FS Bi-directional PHY

## 1.7 Coding and Design Guidelines and Exceptions

The DWC\_otg controller fully complies with the RTL coding practices specified in the *Reuse Methodology Manual*, Third Edition.

## 1.8 Speed and Clock Requirements

The AHB frequency must be greater than 30 MHz. The maximum AHB frequency can vary based on the configuration and the technology library selected. When AHB frequency is less than 30 MHz, the controller cannot meet the five 30-MHz clock USB turnaround requirement with SPRAM (the turnaround time is twelve 30-MHz USB clocks, of which the controller can take only five clocks, with the PHY taking the rest).

When the controller supports periodic channels, you must ensure that you allocate enough system bandwidth on the AHB side. Specifically, you must ensure the following:

- The TxFIFO space is equal to the maximum number of bytes to be transferred in any given microframe. Note that the maximum FIFO depth is capped at 1500 (6000 bytes).
- The minimum RxFIFO space is sum of the highest 2 MPS in any given microframe.
- Packet transfer rate on the AHB is at least 60 Mbytes per second in every microframe which includes latencies and wait states if any.

For more information, see “[Area, Speed, and Power](#)” on page [699](#)

### Effect of System Clocks on Number of Cascaded Hubs Supported by the Device

[Table 1-2](#) lists the number of tiers of hubs that can be supported by the controller operating in device mode based on the PHY clock and AHB clock frequencies, so that the minimum inter-packet gap requirements of the isochronous OUT transactions are met.

**Table 1-2 Number of Hub Levels that the Device can Support Based on System Clock**

Number of Hub Levels	Minimum Inter-Packet Gap at PHY Interface of Device Controller (in HS Bit Times)	Minimum HCLK Frequency Required to Support Worst-Case Inter-Packet Gap (in MHz)
<i>UTMI Data Width = 16</i>		
No Hub	88	110
1	78.4	213
<i>UTMI Data Width = 8</i>		
No Hub	88	30
1	78.4	32
2	68.8	37
3	59.2	44
4	49.6	54
5	40	70



**Note** For devices that use isochronous OUT endpoints across multiple layers of hubs, Synopsys recommends using 8-bit UTMI mode and an AHB clock frequency of 96 MHz or higher.

---

Refer to the “[Impact of System Clocks on Minimum Inter-Packet Gap and Number of Cascaded Hubs Supported by the Device](#)” on page [105](#) and “[Worst-Case Inter-Packet Gap and Maximum Number of Cascaded Hubs Supported by Device Controller](#)” on page [49](#) sections for more details on the impact of the AHB clock and PHY clock frequencies on the following characteristics of the device controller:

- The worst-case inter-packet gap that the device controller can support
- The number of cascaded hubs after which the device can be connected

## 1.9 Power and Area

For power reduction, multiple power rails and clock gating are supported.

Using a standard 0.65µm library, for a configuration with two bidirectional endpoints, gate counts are approximately as follows:

- 35K gates for point-to-point and multi-point (hub and split) applications in AHB Slave mode or External DMA mode
- 40K gates for multi-point (hub and split) applications in Internal DMA mode

For more information, see “[Area, Speed, and Power](#)” on page [699](#).

# 2

## Architecture

---

This chapter describes the design files and hierarchy of the DWC\_otg controller and the various clocks and resets used in the controller. The following topics are discussed:

- “System Level Architecture” on page [60](#)
- “DWC\_otg\_core Architecture” on page [69](#)
- “Power Management Unit Architecture” on page [98](#)
- “System Clocks and Reset” on page [103](#)
- “Interface and Protocol Timing” on page [128](#)
- “USB 1.1 FS Transceiver Interface” on page [139](#)
- “Embedded Host with Micro-AB Receptacle Support” on page [139](#)

## 2.1 System Level Architecture

The DWC\_otg provides the following three architectural modes of operation:

- “Slave-Only Mode”
- “Internal DMA Mode”
- “External DMA Controller Mode” on page 61
- “Host Architecture” on page 65
- “Device Architecture” on page 65

One of these modes is specified for Architecture during coreConsultant configuration. Slave hardware is instantiated even when a DMA mode is selected and, after power-up, software can disable DMA to use Slave mode (and vice versa).

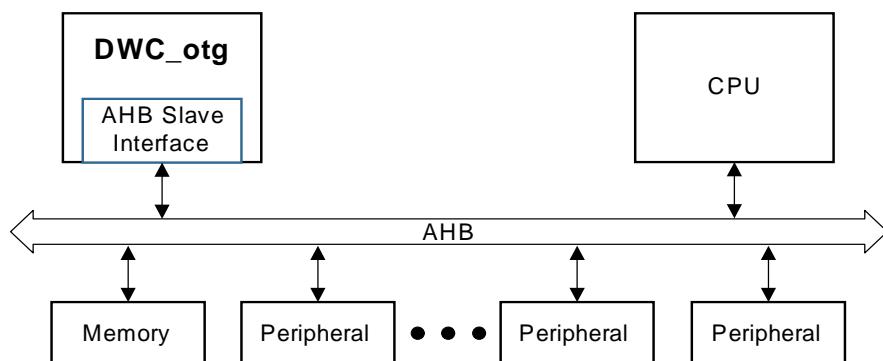


**Note** The DMA and Slave Only modes cannot be changed dynamically. This mode must be selected at the start of initialization and kept static until the controller is reset again.

### Slave-Only Mode

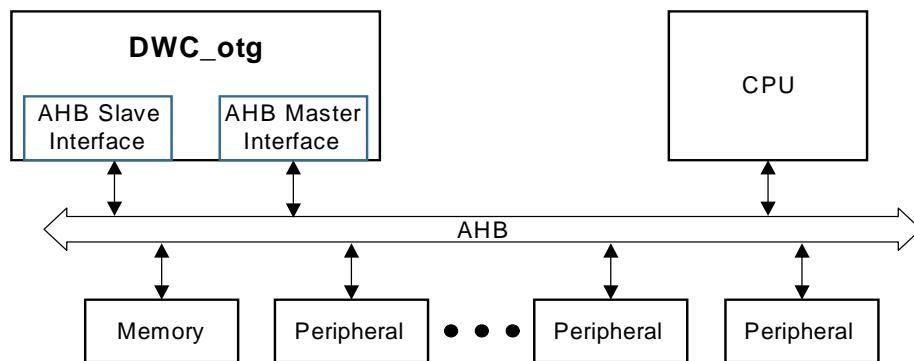
Slave mode is typically selected when area is a concern and you have enough CPU bandwidth to execute the USB driver and to move the data between memory and DWC\_otg.

**Figure 2-1 Slave-Only Mode**



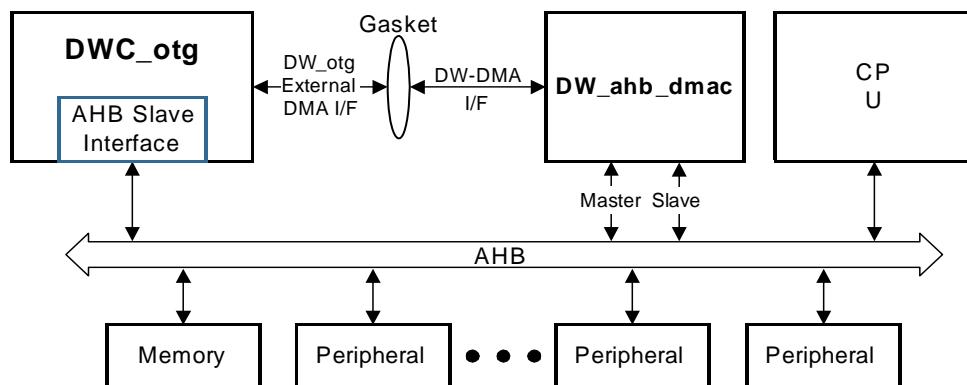
### Internal DMA Mode

Internal DMA mode is typically selected when the CPU bandwidth to process the USB transfer is limited and you would like an internal DMA controller to take care of the data transfers between the memory and DWC\_otg. The driver sets up the transfer and the DWC\_otg interrupts the processor only on transfer completion or an error condition. Internal DMA mode has two modes of operation: Buffer DMA mode and Scatter/Gather DMA mode.

**Figure 2-2 Internal DMA Controller Mode**

### <sup>1</sup>External DMA Controller Mode

External DMA mode is similar to Internal DMA mode in terms of CPU load. This mode is useful when your SoC already has a DMA controller and also has DMA channels available for DWC\_otg. The diagram below shows how the DWC\_otg interfaces to a DW\_ahb\_dmac controller. Depending on the External DMA controller you use, a gasket may be required between the DWC\_otg and the External DMA controller.

**Figure 2-3 External DMA Controller Interface Mode**

The application can perform a USB transaction in Slave or DMA mode. In Slave mode, the data transfer between the host and the system memory is handled by the application. In DMA mode, the data transfer between the OTG controller and the system memory is handled by either an external DMA or the internal DMA.

### 2.1.1 DWC\_otg PMU Module

The DWC\_otg PMU module is responsible for the following functionality:

- Hibernation (complete shutting of the DWC\_otg controller in host and device mode during suspend)
- ADP probing and sensing (available for two configurations - when the ADP controller is external to the controller and when the ADP controller is a part of the controller).

1. This feature is no longer supported, for more details, see [Note](#) on page 36.

Hibernation provides complete power gating for DWC\_otg controller in host and device mode during Suspend. Hibernation can be used in the following types of configurations:

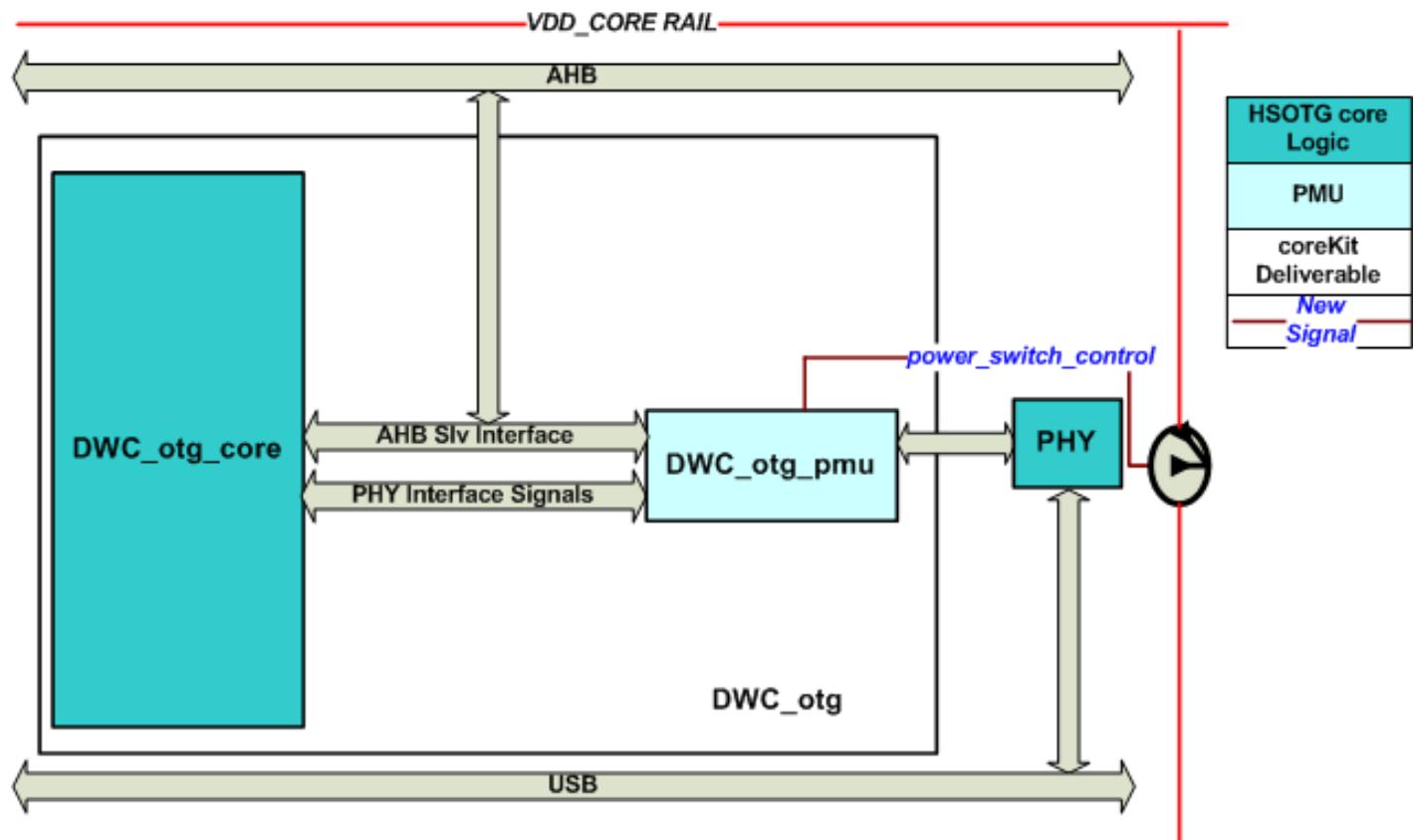
- OTG Mode - Applicable for OTG and Non-OTG Device and Host mode of operation
- Interface Type - UTMI+, ULPI, HSIC, IC\_USB and FS dedicated
- The following architecture types:
  - Internal Buffer Based DMA
  - Internal Descriptor Based DMA
  - <sup>1</sup>External DMA
  - Slave Mode of operation



**Note** The OTG controller and external SPRAM must be on the same power rail to power off the SPRAM.

Figure 2-4 provides a high-level overview of how hibernation works. The DWC\_otg\_pmu module (the Power Management Unit or PMU) is responsible for the hibernation process. For more information about the PMU and the architecture of PMU, see “Power Management Unit Architecture” on page 98.

**Figure 2-4 System Level Block Diagram for PMU**

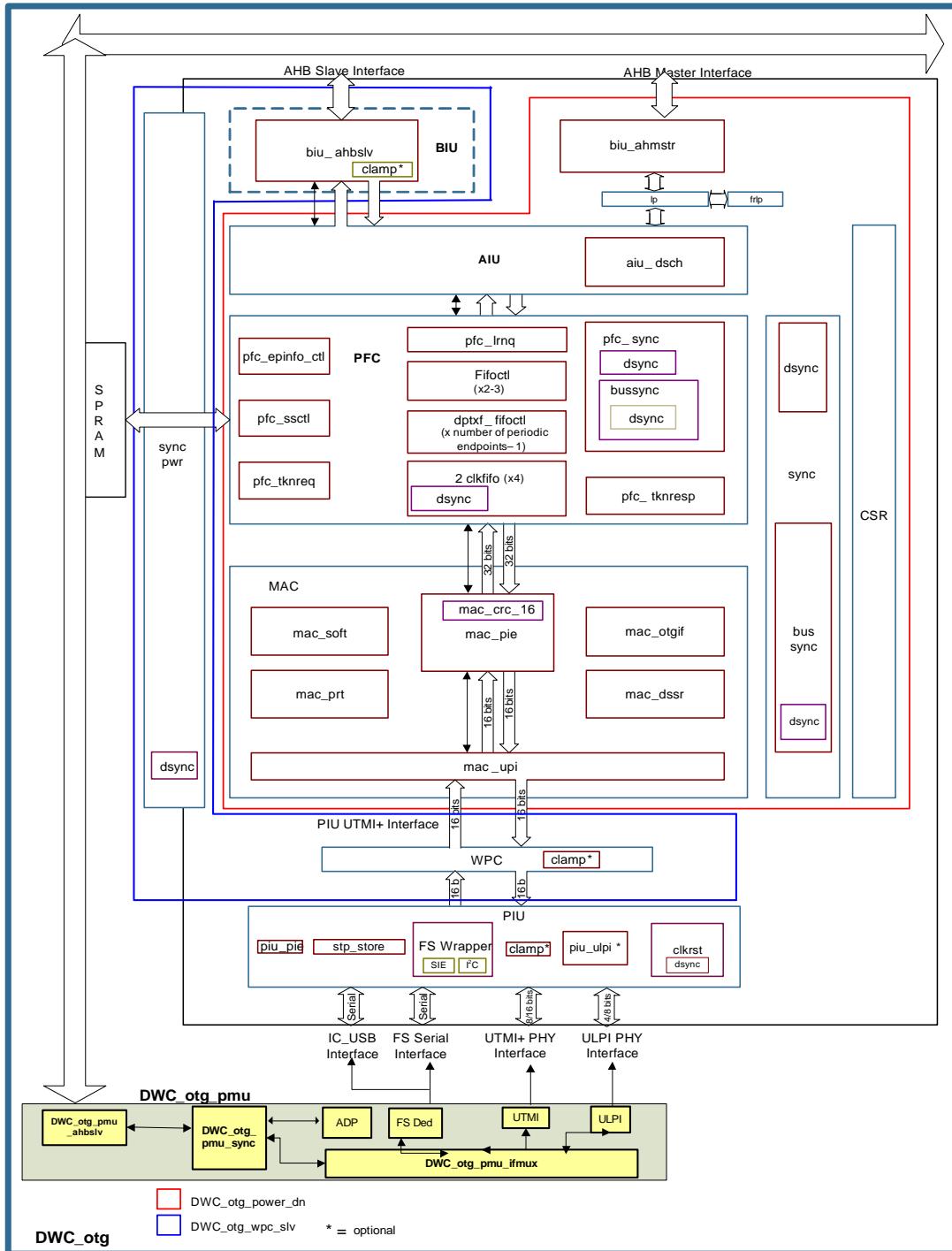


1. This feature is no longer supported, for more details, see [Note](#) on page 36.

## 2.1.2 Design Hierarchy

Figure 2-5 shows the hierarchy of modules in the DWC\_otg controller.

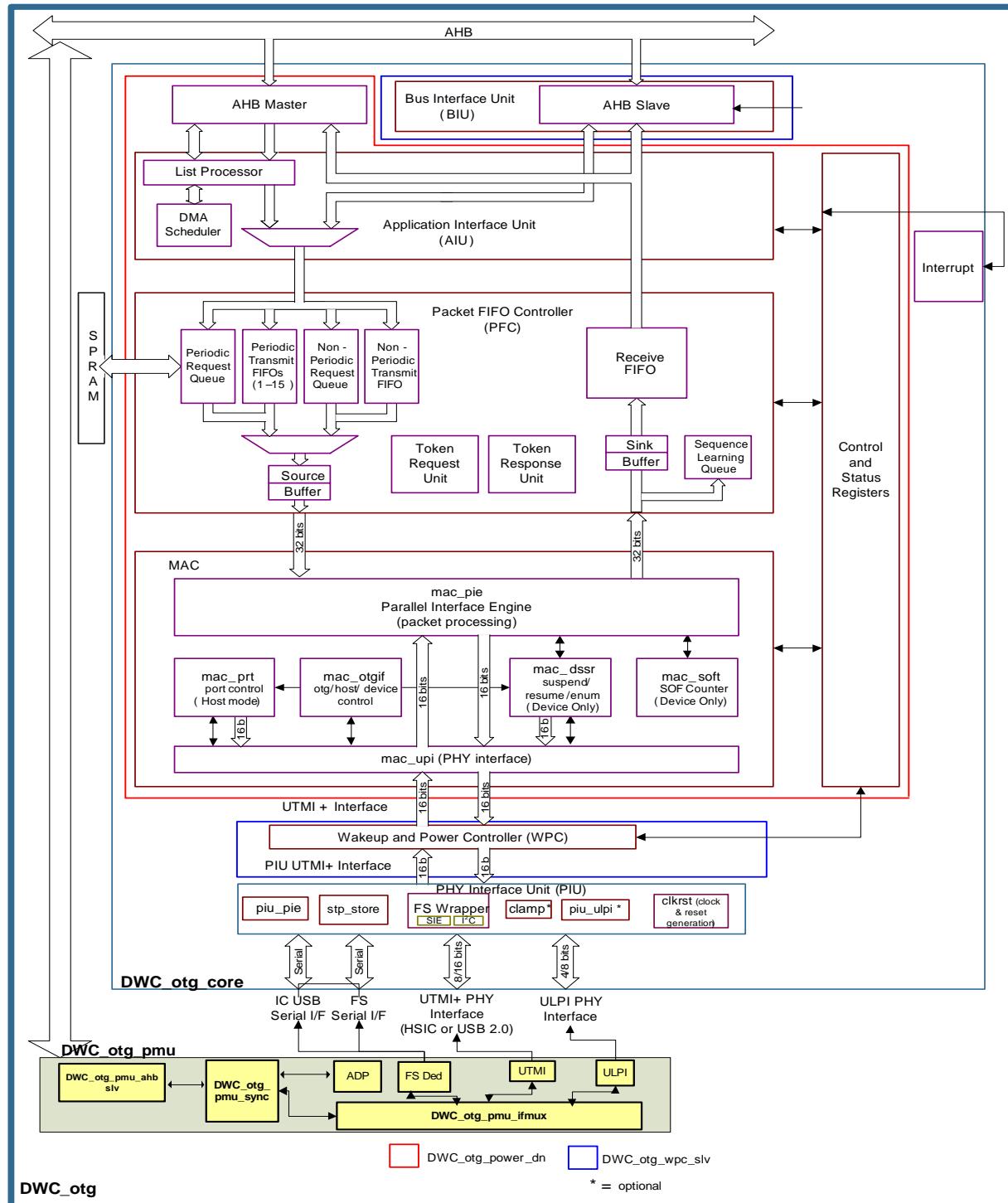
**Figure 2-5 System-Level Module Hierarchy Block Diagram**



## 2.1.3 DWC\_otg Components

Figure 2-6 on page 64 shows the main components and flow of the DWC\_otg system.

**Figure 2-6 System-Level Functional Block Diagram**



## 2.1.4 Host Architecture

The host uses one transmit FIFO for all non-periodic OUT transactions and one transmit FIFO for all periodic OUT transactions (periodic FIFOs 2 to  $n$  are only used in Device mode, where  $n$  is number of periodic IN endpoints in Device mode). 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.

[Figure 2-7](#) on page 67 shows the bus interface architecture of the DWC\_otg controller in Host mode.

## 2.1.5 Device Architecture

This topic describes the DWC\_otg device functionality.

### 2.1.5.1 <sup>1</sup>Shared Transmit FIFO Operation

In Device mode, the controller can be configured to have a shared transmit FIFO for all non-periodic IN endpoints or individual dedicated FIFOs for each IN endpoint. This is controlled by the parameter OTG\_ENDED\_TX\_FIFO during controller configuration.

The shared FIFO architecture provides a low gate count architecture and is recommended when the application does not have more than two IN endpoints. The dedicated transmit FIFO architecture is a more flexible architecture that puts less load on the application than the shared FIFO architecture. In the shared FIFO architecture, the application must predict the order in which the host sends IN tokens and program the endpoints accordingly. Additionally, when packet transmission results in an error, the controller generates an interrupt and the application must re-enable the endpoint. In the dedicated FIFO architecture, the application is not required to predict the host IN token sequence, or errors.

When shared transmit FIFO architecture is used (OTG\_ENDED\_TX\_FIFO=0), the OTG device uses a single transmit FIFO to store the data for all non-periodic endpoints and one transmit FIFO per periodic endpoint to store the data to be transmitted in the next microframe. The data is fetched by the DMA engine or is written by the application, into the transmit FIFOs and is transmitted on the USB when the IN token is received. The Request queue contains the number of the endpoint for which the data is written into the data FIFO.

In Shared FIFO operation, to save area, the Device mode periodic FIFO-1 and the host mode Periodic FIFO are shared (PTxF 1). This is instantiated when either host-mode periodic support or device-mode periodic support is required. In Shared FIFO operation, the Device Mode Periodic FIFOs 2 to  $n$  are conditionally instantiated when the number of Device mode Period IN endpoints are 2 to  $n$ . In dedicated transmit FIFO operation, the FIFOs 2 to  $n$  are instantiated based on the number of IN endpoints (OTG\_NUM\_IN\_EPS).

1. This feature is no longer supported, for more details, see [Note](#) on page 36.

### 2.1.5.2 <sup>1</sup>Order Prediction

In Shared FIFO operation, the application must predict the order in which the USB host accesses the non-periodic endpoints and must write the data into the non-periodic FIFO accordingly. Because each Periodic IN Endpoints has its own FIFO, no order prediction is needed for periodic IN transfers.

### 2.1.5.3 Dedicated Transmit FIFO Operation

When dedicated transmit FIFO architecture is used (`OTG_EN_DED_TX_FIFO = 1`), the controller uses individual transmit FIFOs for each IN endpoint. There are no Request queues associated with any of the FIFOs. There is no need for the application to predict the order in which the USB host is going to access the non-periodic endpoints.

In dedicated transmit FIFO operation, the controller also supports thresholding in the transmit and receive directions when DMA mode is selected. For Transmit mode there are separate controls to enable thresholding for isochronous and non-isochronous transfers. Support is also provided for two different threshold values for the AHB and MAC on the transmit side, and when thresholding is enabled, the controller can be configured to have a less-than-one-packet-sized FIFO. The controller internally handles underrun condition during transmit and corrupts the packet (inverts the CRC) on the USB. During receive with thresholding, when a packet ends up in a FIFO overflow condition, the controller NAKs the OUT packet and internally rewinds the pointers. When thresholding is enabled, the best practice is to have a FIFO size two times the threshold value. If packet transmission results in an underrun condition (eventually resulting in packet corruption on the USB), the host can time out the endpoint after three consecutive errors.

### 2.1.5.4 Single Receive FIFO

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.

Figure 2-7 on page 67 shows the bus interface architecture of the DWC\_otg controller in Host mode.

1. This feature is no longer supported, for more details, see [Note](#) on page 36.

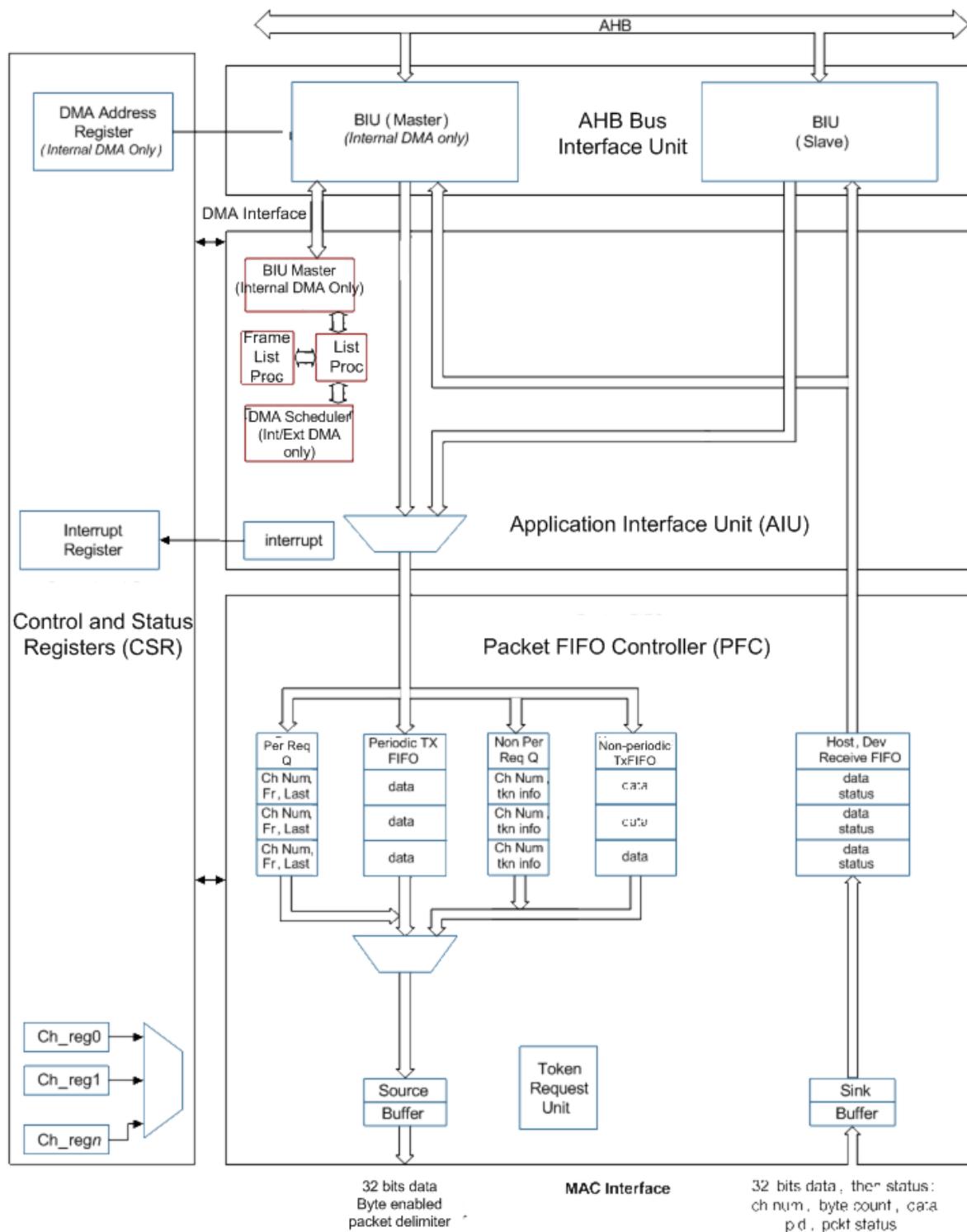
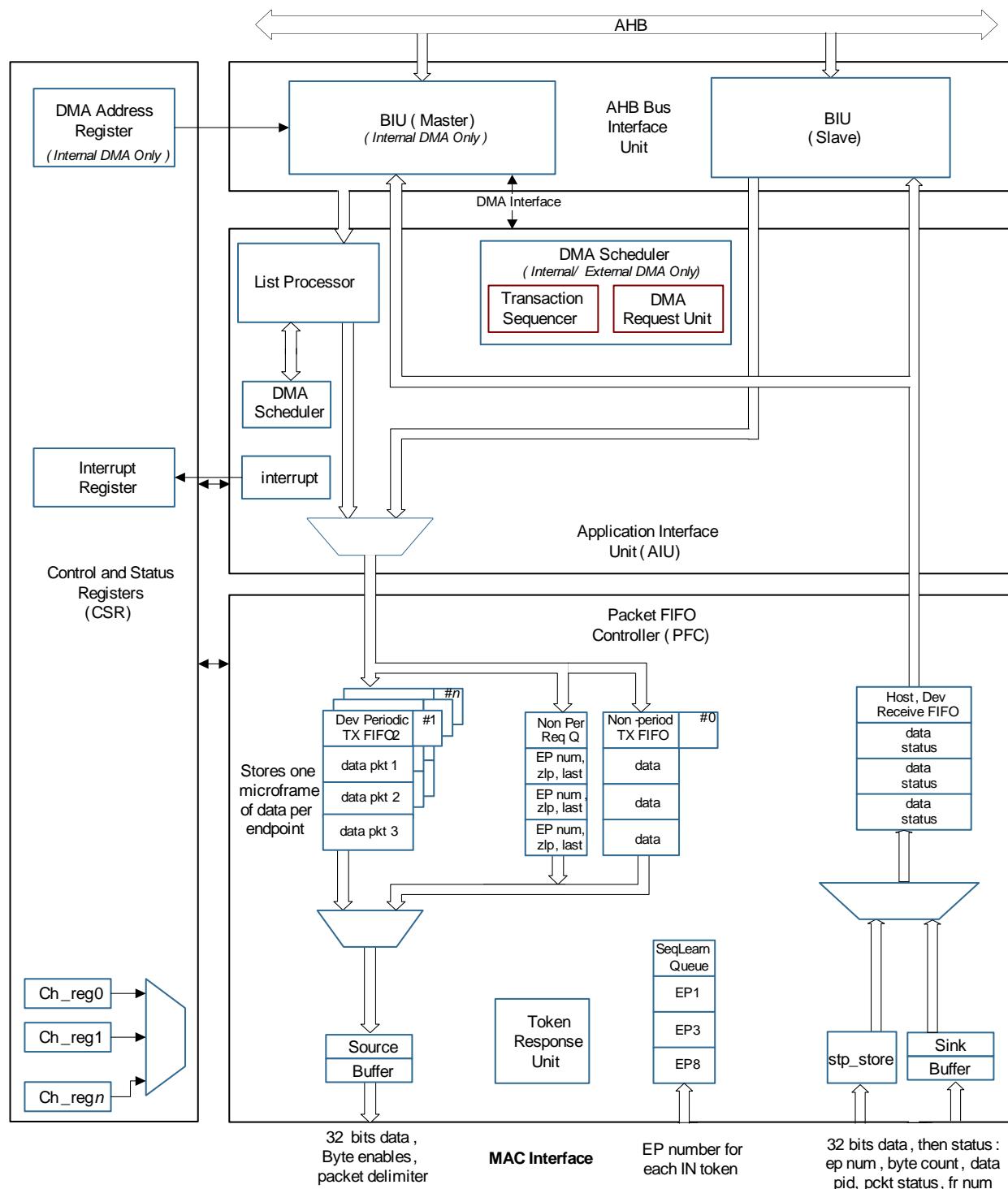
**Figure 2-7 System-Level Host BIU Block Diagram**

Figure 2-8 shows the bus interface architecture of the DWC\_otg controller in Device mode.

**Figure 2-8 System-Level Device BIU Block Diagram**

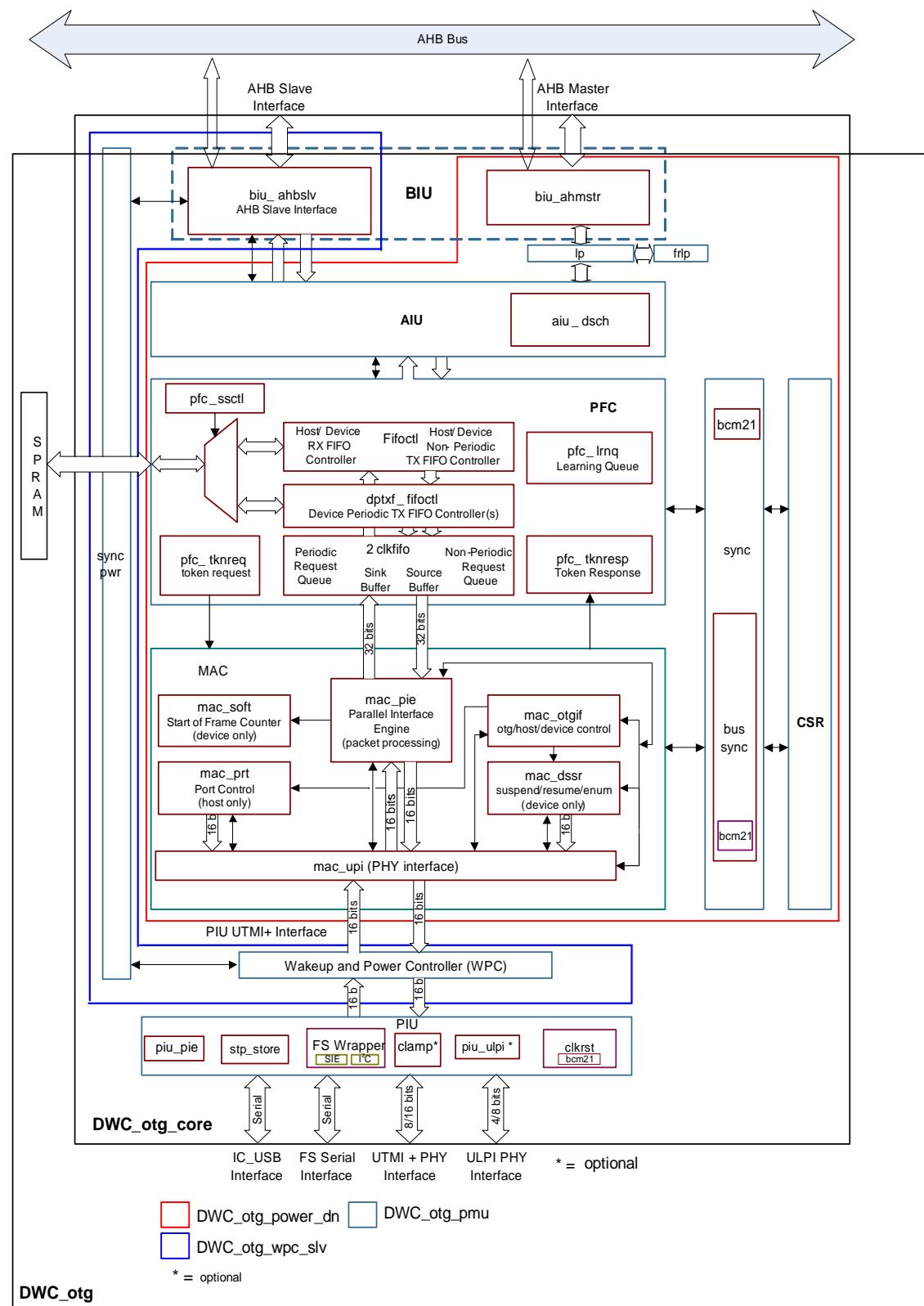


## 2.2 DWC\_otg\_core Architecture

This section describes the DWC\_otg\_core architecture and the major controller components. It also describes how these relate to the system environment.

The OTG controller shares a single SPRAM between transmit (periodic and non-periodic) and receive FIFOs. The size of each FIFO (transmit and receive) can be programmed dynamically when <sup>1</sup>Enable Dynamic FIFO Sizing? is selected during coreConsultant configuration. [Figure 2-9](#) on page [70](#) shows the DWC\_otg\_core blocks and their functions.

1. This option is no longer supported, for more details, see [Note](#) on page [36](#).

**Figure 2-9 DWC\_otg\_core Architecture**

## 2.2.1 AHB Bus Interface Unit (BIU)

The AHB Bus Interface Unit instantiates the following modules:

- “[AHB Slave Bus Interface Unit \(BIUS\)](#)”
- “[AHB Master Bus Interface Unit \(BIUM\)](#)” on page [78](#)

### 2.2.1.1 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. DFIFO read/write access is made available only for testing purposes. The read, write, push, and pop signals are active high for one clock. The wait states for different accesses are:

- Zero wait states on writes to the CSR, except when the state machine is busy with a previous push to DFIFO/Write Data RAM
- Zero wait states on pushes to DFIFO, one additional wait state when the USB accesses the FIFO RAM
- Zero wait states on debug writes to DFIFO, one additional wait state when the USB accesses the FIFO RAM
- One wait state on reads to CSR registers
- One initial wait state on NSEQ pop access and no subsequent wait states on NSEQ/SEQ back-to-back pops to DFIFO; one additional wait state when the USB accesses the FIFO RAM.
- Two wait states on software debug reads to DFIFO, one additional wait state when the USB accesses the FIFO RAM (debug access to DFIFO is not optimized to save area).

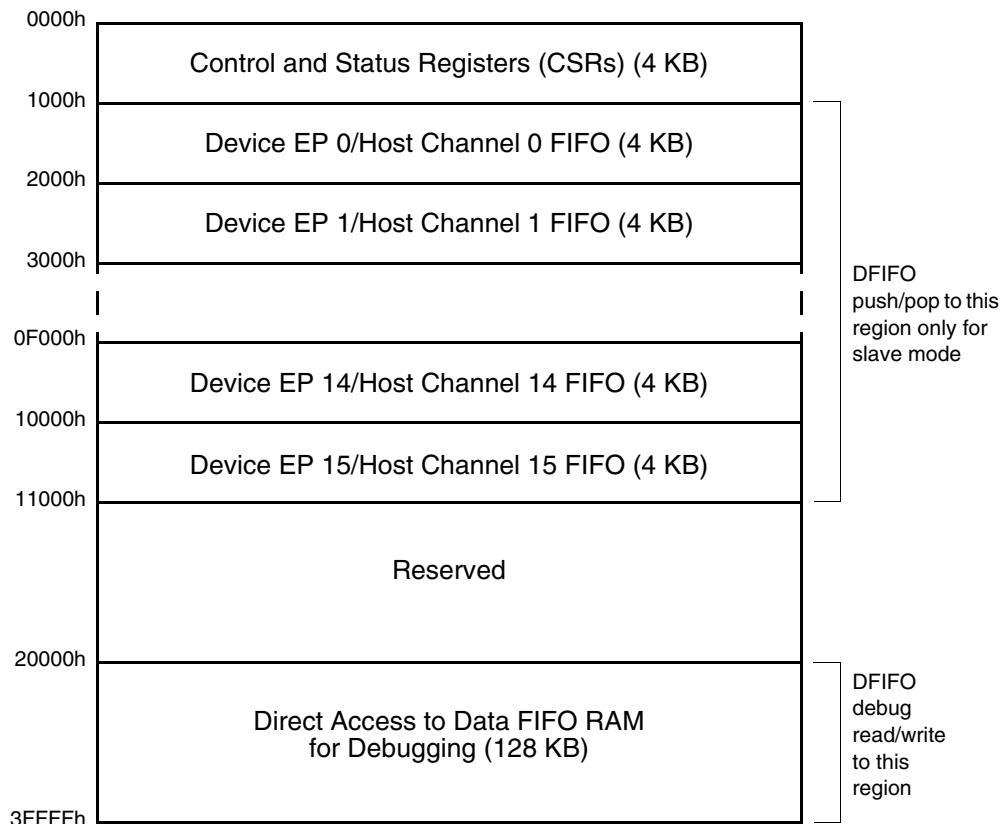
Characteristics of this interface include the following:

- Fully AMBA 2.0-Compliant AHB Slave—No restrictions
- Supports INCR4, INCR8, INCR16, INCR and SINGLE transfers
- Supports busy and early terminations
- CSR and DFIFO reads/writes must always be 32-bit; 8- and 16-bit write accesses commit unknown values to the DFIFO.
- Generates OKAY responses only
- Does not generate SPLIT, RETRY, or ERROR responses

## Address Map

[Figure 2-10](#) displays the address map of the controller.

**Figure 2-10 Address Map**



Note the following:

- s\_haddr[17:12]=0 is mapped to CSR (4 KB region, offset 'h0')
- (s\_haddr[17]=1'b0 & s\_addr[16:12]!= 0) is mapped to Data FIFO push/ pop access. 4K is allotted each device endpoint/host channel. Even though only one address is enough to push/ pop to a device endpoint or host channel, 4K is allotted to each endpoint or channel to allow external DMA controllers to perform SEQ AHB burst access to DFIFO region.
- s\_haddr[17]=1'b1 is mapped for DFIFO debug access. One can directly access the DFIFO without having to push/ pop through the FIFOs.
  - The Application can read the DFIFO Contents (RxFIFO and Tx FIFO) for Debug purposes in Host and Device mode for Slave and DMA architecture.
  - The DFIFO contents can be read only when the RxFIFO is empty, all Channels for Host and all EPs are disabled for Device.

- c. These conditions need to be met so that the MAC side and AHB (DMA) side of the controller stop pushing from or pushing to the FIFO when the SW is trying to access DFIFO for debug purposes.
- d. For Device Mode DCTL.SGOUTNak must be set to prevent any new packets/data (except SETUP) from MAC side being written into the FIFO effect in the core.
- e. Even when the FIFO is empty, the contents written previously are still valid until they are overwritten by either the MAC or AHB side of the DFIFO.



**Note** Debug access is not meant to be used when the controller is in normal mode of operation and is expected to be used only for debug purpose.

- 'h1C and 'h20 addresses are additionally used for DFIFO status reads and pops to DFIFO, bypassing CSR access.

The AHB Slave interface also has a Power Optimization Mode Control register mapped to address 'hE00. Remote wakeup, session request, and device disconnect interrupt and mask bits are also implemented in this module. PHY selection control bits are also implemented in this block, not in the CSRs, because the CSR module is powered down during suspends.

To make the software transparent to the hardware implementation, one 4 KB region is mapped to each device endpoint or host channel. The controller has:

- One common RxFIFO, used in Host and Device modes
- <sup>1</sup>One common Non-periodic Tx FIFO used in Host and Device modes when OTG\_ENDED\_TX\_FIFO = 0
- One common Periodic TxFIFO, used in Host mode
- One Periodic Tx FIFO for each periodic IN endpoint in Device mode when <sup>2</sup>OTG\_ENDED\_TX\_FIFO = 0.
- Separate IN endpoint transmit FIFO for each Device mode IN endpoints in Dedicated Transmit FIFO operation (OTG\_ENDED\_TX\_FIFO = 1)
- The FIFO SPRAM is also used for storing some register values to save gates. In Scatter/Gather DMA mode, four SPRAM locations (four 35-bit words) are reserved for this. In DMA and Slave modes (non-Scatter/Gather mode), one SPRAM location (one 35-bit word) is used for storing the DMA address.

Within the controller:

- Read access to any one of the 4 KB regions is mapped to the RxFIFO.
  - Writes to any non-periodic IN endpoint or OUT channel are mapped to the Non-periodic TxFIFO.
  - In Host mode, writes to any periodic OUT channel are mapped to the common Periodic TxFIFO.
  - In Device mode, write access to a Periodic IN endpoint (or all IN endpoints in Dedicated FIFO operation) is mapped to the corresponding endpoint Periodic (periodic/non-periodic in Dedicated Transmit FIFO Operation) Tx FIFO (bits 30:27 of the Device Endpoint Control Register maps an endpoint to a specific device periodic FIFO).
1. This option is no longer supported, for more details, see [Note](#) on page 36.
  2. This option is no longer supported, for more details, see [Note](#) on page 36.

<sup>1</sup>In Shared FIFO operation, an IN INTERRUPT endpoint can be mapped to either of two options:

- The common Non-periodic Tx FIFO
- Assigned to a separate Periodic Tx FIFO

When DIEPCTLn.PTxFNum is 0, the endpoint is mapped to common Non-periodic Tx FIFO, otherwise it is mapped to the FIFO number selected by these fields.

Hardware maintains the Non-periodic Tx queue and Host mode Tx Periodic queues for internal operation. For debugging, software can read the top of the queue information. Because these queues' read domain is in the PHY domain, no debug pop access is provided to these queues, saving area; access is provided only to the top of the queue.



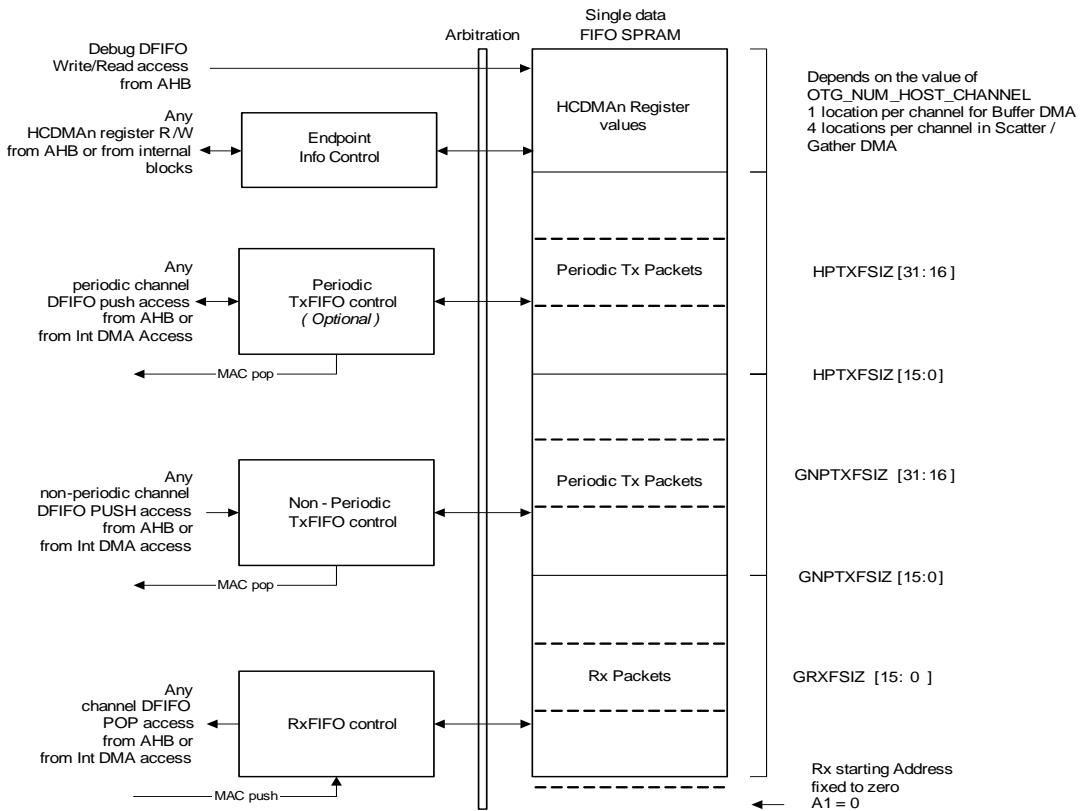
**Note** In Device mode, each periodic IN endpoint has a separate buffer allocated, holding only one packet at a time. Therefore, there is no Device mode Periodic Queue. When OTG\_ENDED\_TX\_FIFO = 1, there is no queue associated with any device IN transmit FIFO. Limiting to one packet shrinks the individual FIFO controller, saving area.

---

1. This option is no longer supported, for more details, see [Note](#) on page 36.

Figure 2-11 shows FIFO mapping in Host mode.

### Figure 2-11 Host Mode FIFO Address Mapping and AHB FIFO Access Mapping



When Dynamic FIFO Sizing is disabled, the values in these registers are constants, derived from cC parameters as

```

GRXFSIZ[31:15] = OTG_RX_DFIFO_DEPTH
GNPTXFSIZ[15:0] = OTG_RX_DFIFO_DEPTH
GNPTXFSIZ[31:16] = OTG_TX_NPERIO_DFIFO_DEPTH (Shared FIFO operation)
GNPTXFSIZ[31:16] = OTG_TX_HPERIO_DFIFO_DEPTH (Dedicated FIFO operation)
HPTXFSIZ[15:0] = OTG_RX_DFIFO_DEPTH + OTG_TX_NPERIO_DFIFO_DEPTH (Shared FIFO Operation)
HPTXFSIZ[15:0] = OTG_RX_DFIFO_DEPTH + OTG_TX_HPERIO_DFIFO_DEPTH (Dedicated FIFO Operation)

```

The formula for HPTXFSIZ[31:16] is:

```

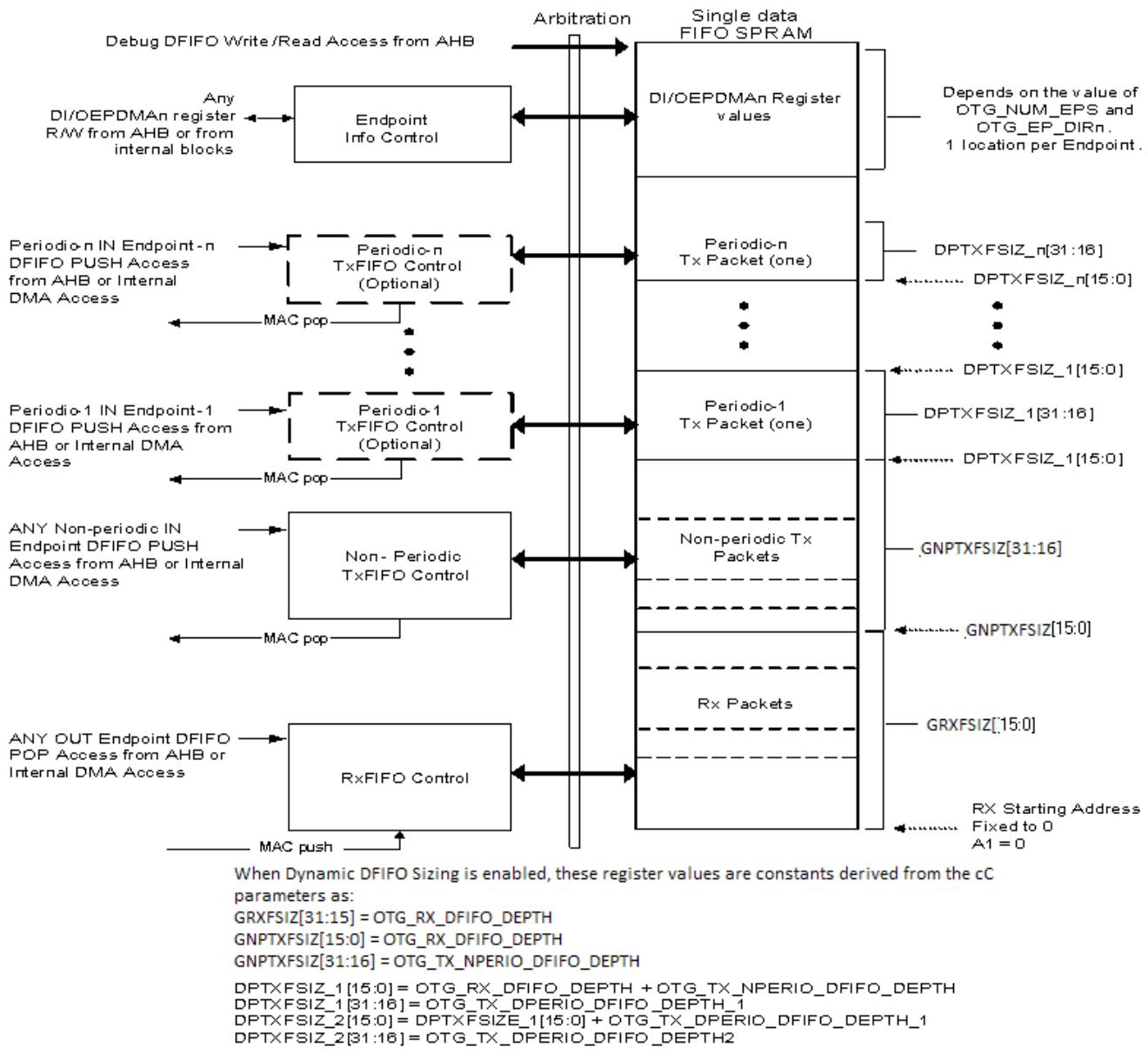
(OTG_EN_PERIO_HOST==1) && (((OTG_EN_DED_TX_FIFO==0) ? OTG_NUM_PERIO_EPS :
(OTG_NUM_IN_EPS - 1)) == 0) || (OTG_MODE == 5) || (OTG_MODE == 6)) ?
OTG_TX_HPERIO_DFIFO_DEPTH :
(OTG_EN_PERIO_HOST==0) && ((OTG_MODE == 3) || (OTG_MODE == 4)) && (((OTG_EN_DED_TX_FIFO==0) ?
OTG_NUM_PERIO_EPS : (OTG_NUM_IN_EPS - 1)) > 0) ?
((OTG_EN_DED_TX_FIFO==0) ? OTG_TX_DPERIO_DFIFO_DEPTH_1 : OTG_TX_DINEP_DFIFO_DEPTH_1) :
(OTG_TX_HPERIO_DFIFO_DEPTH >= ((OTG_EN_DED_TX_FIFO==0) ? OTG_TX_DPERIO_DFIFO_DEPTH_1 :
OTG_TX_DINEP_DFIFO_DEPTH_1)) ?
OTG_TX_HPERIO_DFIFO_DEPTH :
((OTG_EN_DED_TX_FIFO==0) ? OTG_TX_DPERIO_DFIFO_DEPTH_1 : OTG_TX_DINEP_DFIFO_DEPTH_1)

```

Note:  
When the host is operating in Internal DMA mode, the last locations of the SPRAM are used to store the DMAADDR values for each channel.

Figure 2-12 shows FIFO mapping in Device mode in Shared FIFO operation.

**Figure 2-12 1Device Mode FIFO Address Mapping and AHB FIFO Access Mapping (Shared FIFO)**



1. This option is no longer supported, for more details, see [Note](#) on page 36.

Figure 2-13 shows FIFO mapping in Device mode.

### Figure 2-13 Device Mode FIFO Address Mapping and AHB FIFO Access Mapping (Dedicated FIFO)

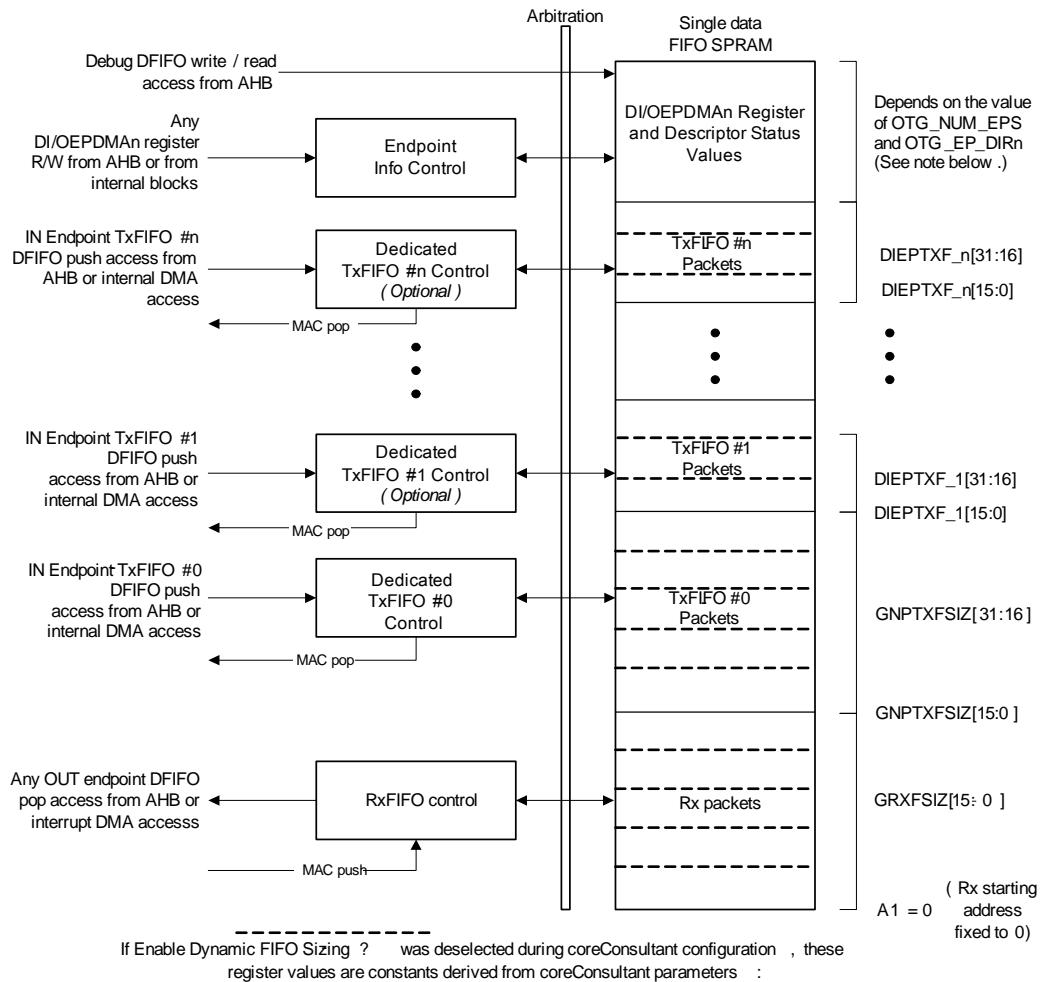
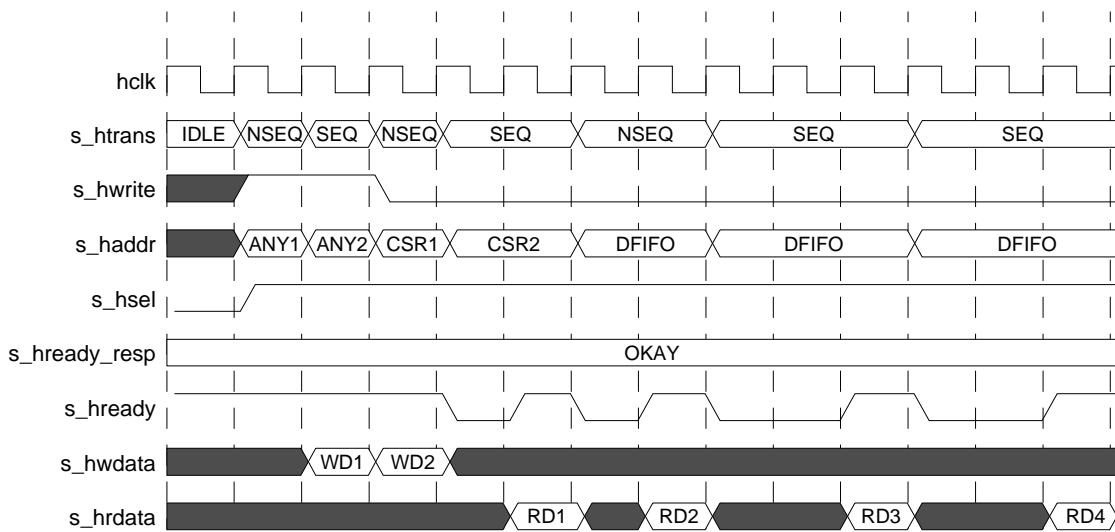


Figure 2-14 shows AHB Slave interface timing.

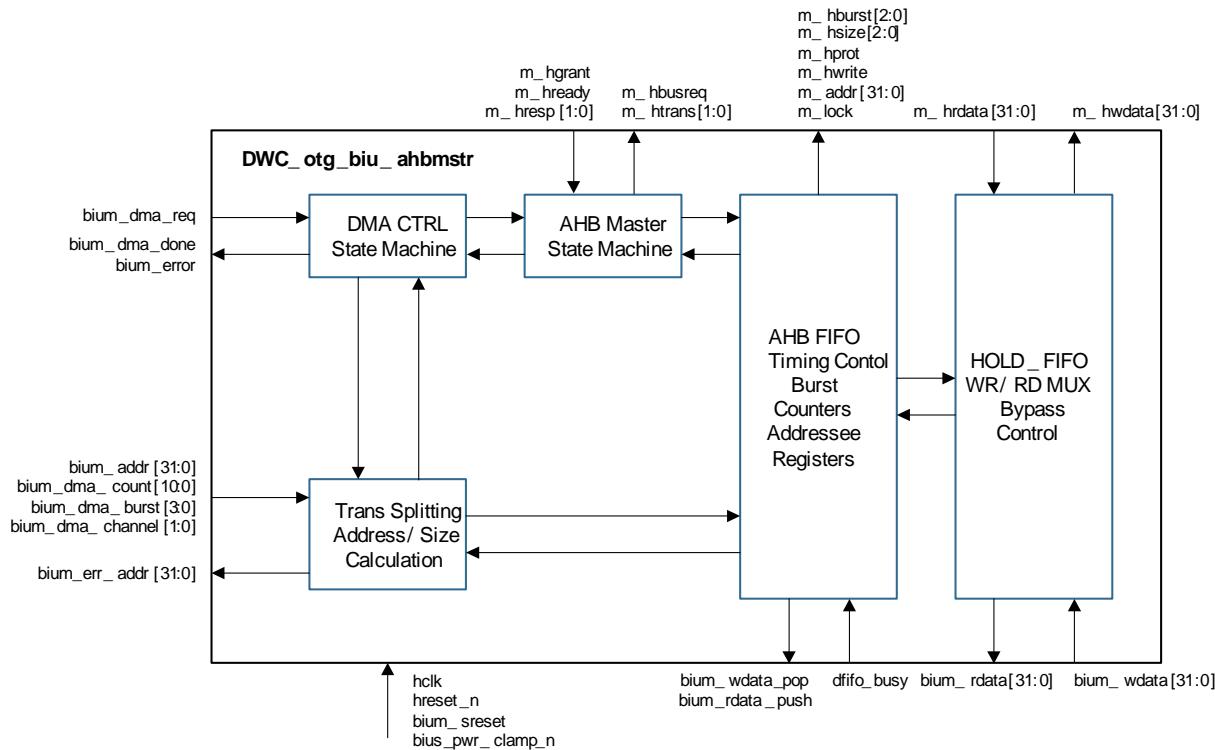
**Figure 2-14 AHB Slave Interface Timing**



### 2.2.1.2 AHB Master Bus Interface Unit (BIUM)

The Internal DMA controller translates internal DMA requests into AHB Master requests. The DMA address, transfer count, and packet count registers reside in the CSR block. The selected channel's address is given as input to the Internal DMA controller. Figure 2-15 shows the AHB Master interface. See “[External DMA Controller Mode](#)” on page 61 for external DMA interface details.

**Figure 2-15 AHB Master Block Diagram**



The AHB Master interface unit converts the internal DMA request cycles into AHB cycles. Characteristics of this interface include the following:

- Fully AMBA 2.0-compliant AHB Master – No restrictions
- You can choose SINGLE, INCR, INC4, INC8, or INC16 burst types. When INC4, INC8, or INC16 is chosen, SINGLE and INCR cycles can happen due to a split, retry, or early termination. The DWC\_otg controller uses INCR in the following scenarios:
  - Remaining Data Transfers (In case the data length is not a direct multiple of the burst length the remaining data is sent using INCR burst type).
  - Data Transfer after SPLIT
  - Data Transfer after RETRY



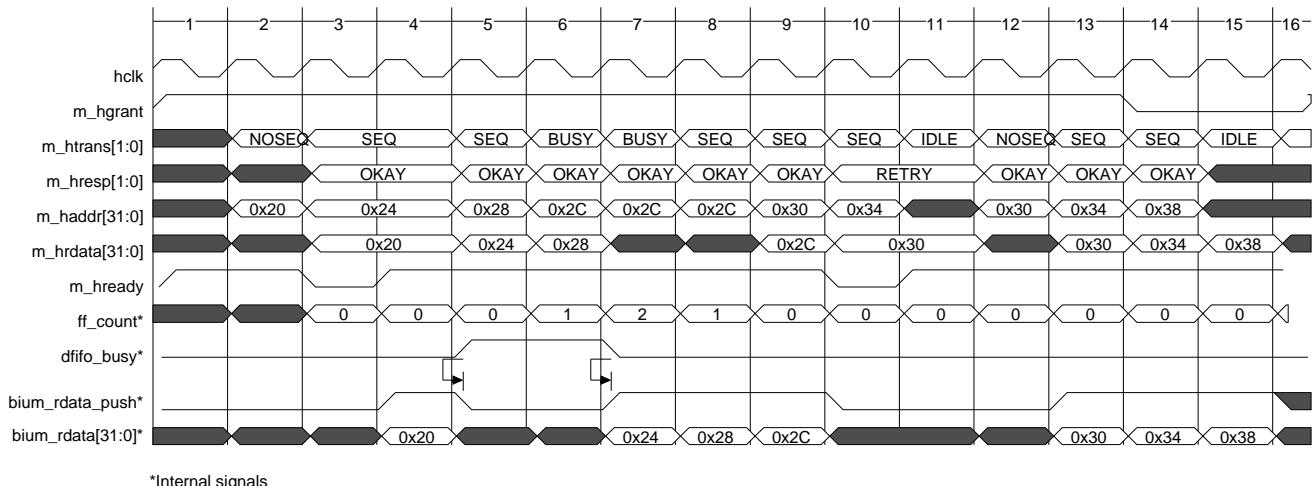
**Note** The application can select whether it wants the remaining data transfer to be completed using INCR or Single burst type.

- Handles AHB SPLIT, RETRY, ERROR conditions and early termination
- Inserts BUSY cycle when needed by application
- Handles AHB 1 KB boundary breaking
- AHB master's data bus and address bus are 32 bits wide. The DWC\_otg controller generates transfer with hsize=2 (DWORD) and 0 (BYTE).

The AHB Master interface uses the same dma\_req and dma\_done handshake signals used in the External DMA interface. When the Internal DMA controller is tested, the External DMA interface is thus also automatically tested. In addition to the External DMA interface signals, this block also has Internal DMA controller-specific signals. This block breaks DMA requests into multiple AHB cycles and transfers data between the DMA and the system memory.

Figure 2-16 shows the timing when the BIUM transfers data from the AHB to the packet FIFO.

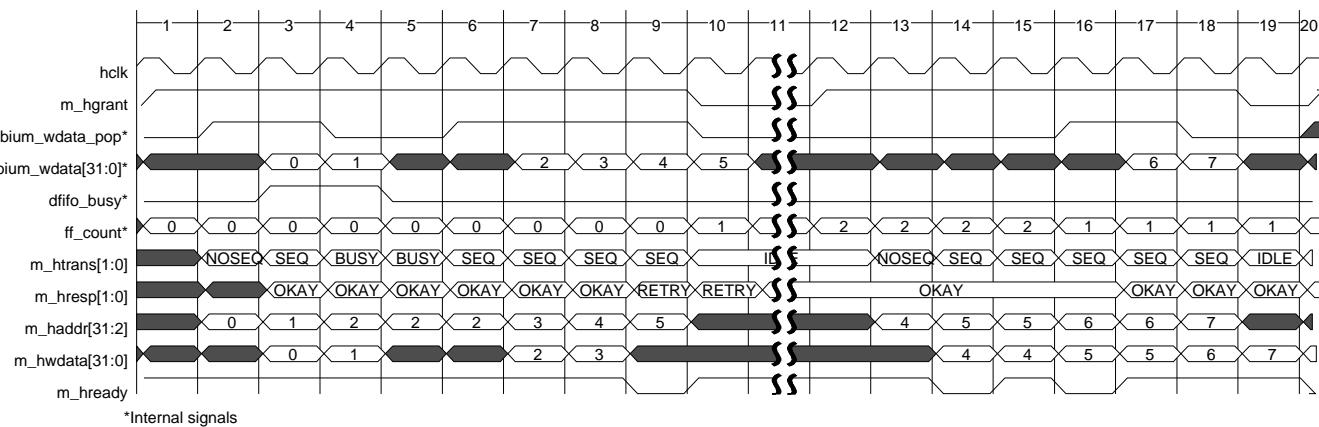
**Figure 2-16 AHB Master Interface Timing for Reads**



\*Internal signals

Figure 2-17 shows the timing diagrams when the BIUM transfers data from the packet FIFO to the AHB.

**Figure 2-17 AHB Master Interface Timing for Writes to the AHB**



Note the following points regarding AHB write transfer timing:

- A 2-level-deep hold FIFO is put between the AHB and the packet FIFO. When the AHB Slave is not ready ( $m\_hready = 0$ ), the packet FIFO output data is written to the hold FIFO. The  $ff\_count$  signal shows how the FIFO count changes.
- The  $dfifo\_busy$  signal is registered before being used to generate AHB Master BUSY cycle.
- The data popped from the packet FIFO is always put in the hold FIFO. AHB write data is always read from the hold FIFO. Thus, the output is always registered.
- For the AHB Master, split transfers are not different from retry transfers; however, the AHB arbiter behaves differently for each transfer type.

## 2.2.2 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 Controller Interrupt register (GINTSTS). These register bits must be active in Power Down mode, and hence are implemented in the BIU Slave module. The AHB Slave Interface unit can write and read registers, and the controller can set or reset the interrupt and status bits. The interrupts have priority over AHB Slave Interface updates. For example, when an AHB Slave Interface access tries to clear an interrupt bit when the controller is setting the same interrupt bit, the interrupt bit is set and the AHB write to that bit is ignored.

Because the controller does not check byte enables, all register accesses are 32 bits.

Even though AHB Slave writes have only the granularity of byte writes, each interrupt bit in an interrupt register is controlled separately. Thus, the AHB Slave interface can clear an interrupt bit when the controller is setting a different interrupt bit, but no read modify byte write (which could cause erroneous interrupt behaviors) is performed. The following example code shows how interrupt register bits are coded:

```
always @ (posedge hclk_int)
  if(int_src[0])
    int[0] <= 1'b1; // set by interrupt source
  else if(bius_write_csr & (bius_addr[11:2] == 'INT_REG_ADDR) & bius_wdata[0])
    int[0] <= 1'b0; // reset by AHB write
```

```

if(int_src[1])
    int[1] <= 1'b1;
else if(bius_write_csr & (bius_addr[11:2] == 'INT_REG_ADDR) & bius_wdata[1])
    int[1] <= 1'b0;
.....
if(int_src[31])
    int[31] <= 1'b1;
else if(bius_addr[11:2] == 'INT_REG_ADDR & bius_wdata[31])
    int[31] <= 1'b0;

```

All implemented registers are 32 bits wide, and only those registers required by the configuration are implemented. For example, when the controller is configured for only two endpoints, then registers corresponding to endpoints 3–15 are not implemented. In addition, because the controller can be only be either in Host or Device mode at a given time, the host and device registers use the same physical registers to save area. Writes to unimplemented registers are ignored, and reads from them return an unknown value. Cycles complete gracefully with an OK response. When Device mode registers are accessed during Host mode and vice-versa, the Mode Mismatch interrupt (GINTSTS.ModeMis) are set, write data is ignored, and read data returns an unknown value.

### 2.2.3 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

The AIU is responsible for the following functions.

1. Generating and writing the delimiter (byte enables and last DWORD indicator) into the transmit FIFOs (based on FIFO number) for transmit packets in Device (IN) and Host (OUT) modes
2. Writing a token into the Request Queue (periodic/non-periodic) for transmit transactions in Device and Host modes
3. Writing a token into the Request Queue for receive (IN) transactions (only in Host mode)
4. Selecting the Periodic or Non-periodic Request Queue based on the type of transaction. The Periodic Queue is selected when the channel or endpoint type indicates
5. Generating interrupts for both transmit and receiving operations in Device and Host modes
6. Address decoding for selecting the appropriate transmit FIFOs and queues
7. Generating DMA interface signals to access an internal or external DMA controller for data transfer.
8. Round-robin arbitration among periodic and non-periodic channels and endpoints. Higher priority is always given to periodic channels or endpoints
9. Transmit/receive threshold handling when Tx thresholding is enabled.
10. Rewinding logic for DMA address and transfer size during an Underrun/Overflow condition when thresholding is enabled.

### 2.2.3.1 DMA Scheduler (DSCH)

The DMA Scheduler is an optional block is responsible for interfacing to an external or internal DMA. This block is used only in DMA mode. It controls the transfer of data packets between the system memory and the OTG controller for both Internal and External DMA. The DMA scheduler includes the following:

- Arbiter

This logic provides the sequence in which the channels/endpoints are to be processed in DMA mode. In Host mode, the arbiter provides round-robin arbitration among periodic and non-periodic channels. Periodic channels are processed with higher priority. In Device mode with Shared FIFO operation (OTG\_ENDED\_TX\_FIFO = 0), the arbiter provides a round-robin arbitration only among periodic endpoints. Arbitration for non-periodic endpoints is based on Next EP Number<sup>1</sup> link register values. Periodic endpoints are processed with higher priority.

In Device mode, during a Dedicated Transmit FIFO operation (OTG\_ENDED\_TX\_FIFO = 1), and when threshold is enabled, the priority is as follows:

- a. Any transmit endpoint which is active on the USB.
- b. Any receive data in a receive FIFO.
- c. Round-robin arbitration on periodic transmit endpoints.
- d. Round-robin arbitration on non-periodic transmit endpoints.

In device mode when dedicated FIFO mode is used and when thresholding is not enabled, round robin arbitration is used for periodic and non-periodic IN endpoints with priority given to periodic IN endpoints.

- DMA Request State Machine

The state machine is responsible for the following:

- ❑ Requesting the External/Internal DMA for data fetch (from system memory to transmit FIFO, one maximum packet size or last packet size at a time)
- ❑ Writing the OUT request token into the request queue at the end of data fetch, in host mode and in Device mode when OTG\_ENDED\_TX\_FIFO = 0.
- ❑ Writing the IN request token into the Request Queue in Host mode
- ❑ Requesting the External/Internal DMA for data update (from receive FIFO to system memory, one maximum packet size or last packet size at a time)
- ❑ Writing the Request Queue for ping, complete split, zero-length packet, or disable channel requests for the host.

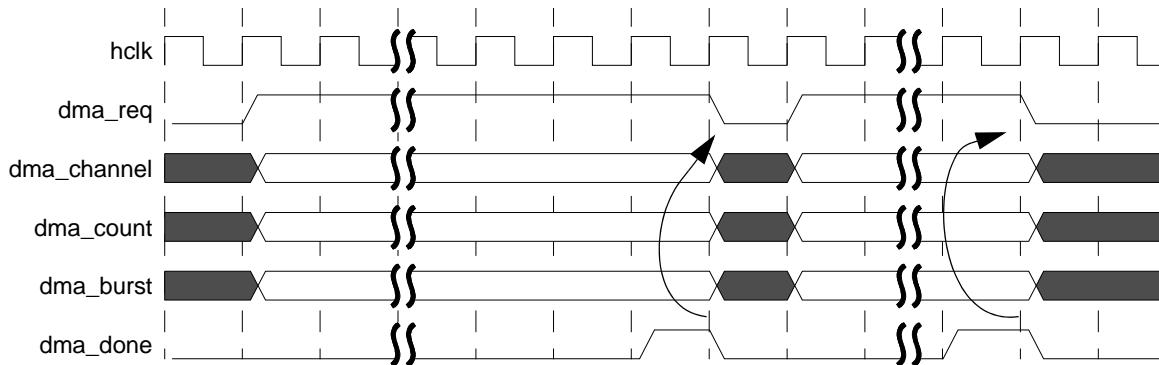
1. The Next EP Number field must point to a valid linked endpoint always, even when the endpoint is not enabled.

### 2.2.3.2 <sup>1</sup>External DMA Controller Interface

The AIU generates the external DMA interface signals. Depending upon your DMA controller, you could need additional external logic. This is not supported in Dedicated Transmit FIFO operations (OTG\_ENDED\_TX\_FIFO = 1).

Figure 2-18 shows the External DMA Interface signals.

**Figure 2-18 External DMA Interface Timing**



1. This feature is no longer supported, for more details, see [Note](#) on page 36.

## 2.2.4 Packet FIFO Controller (PFC)

Figure 2-19 represents the Packet FIFO controller during a Shared FIFO operation.

**Figure 2-19 Packet FIFO Controller (Shared FIFO)**

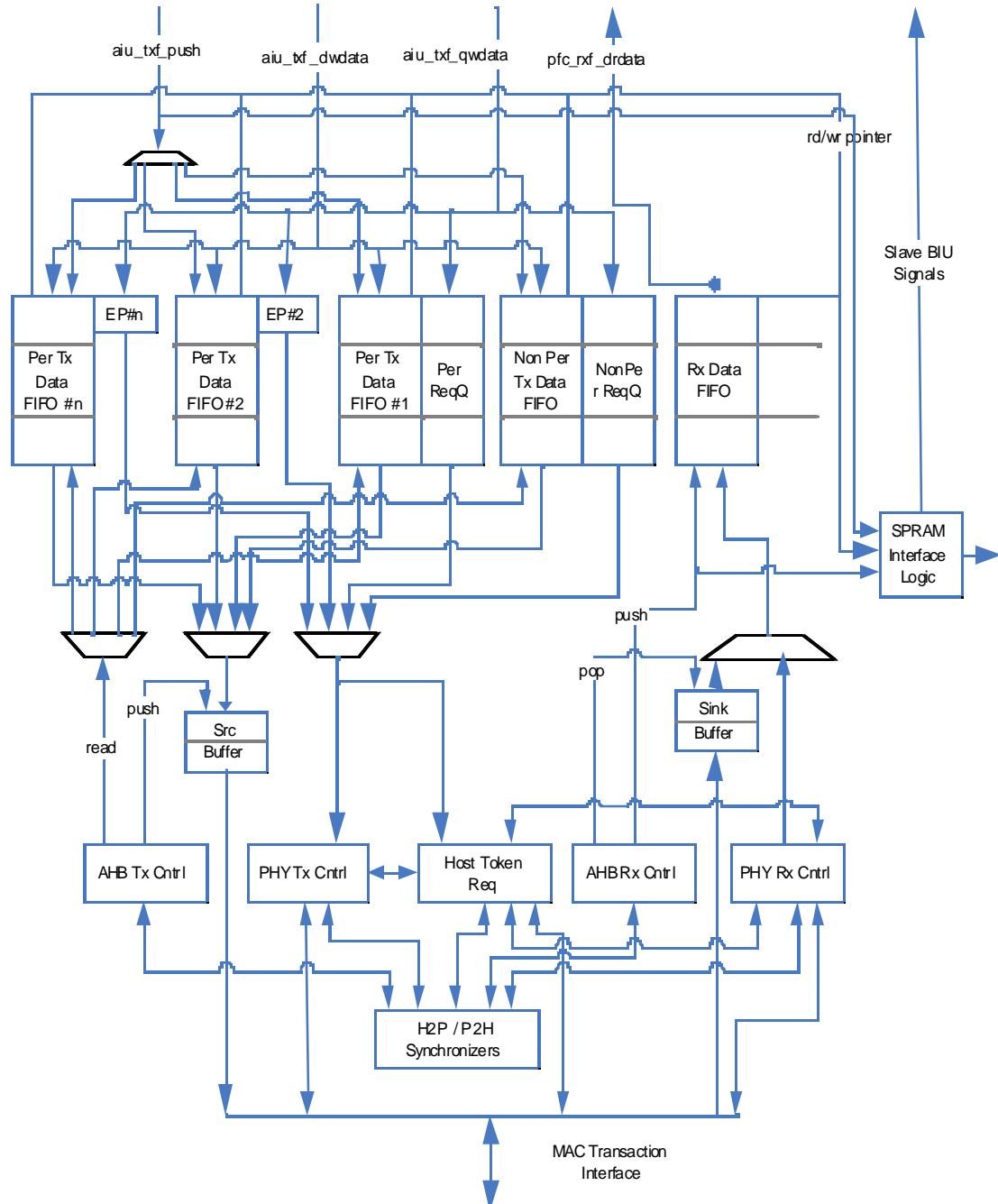
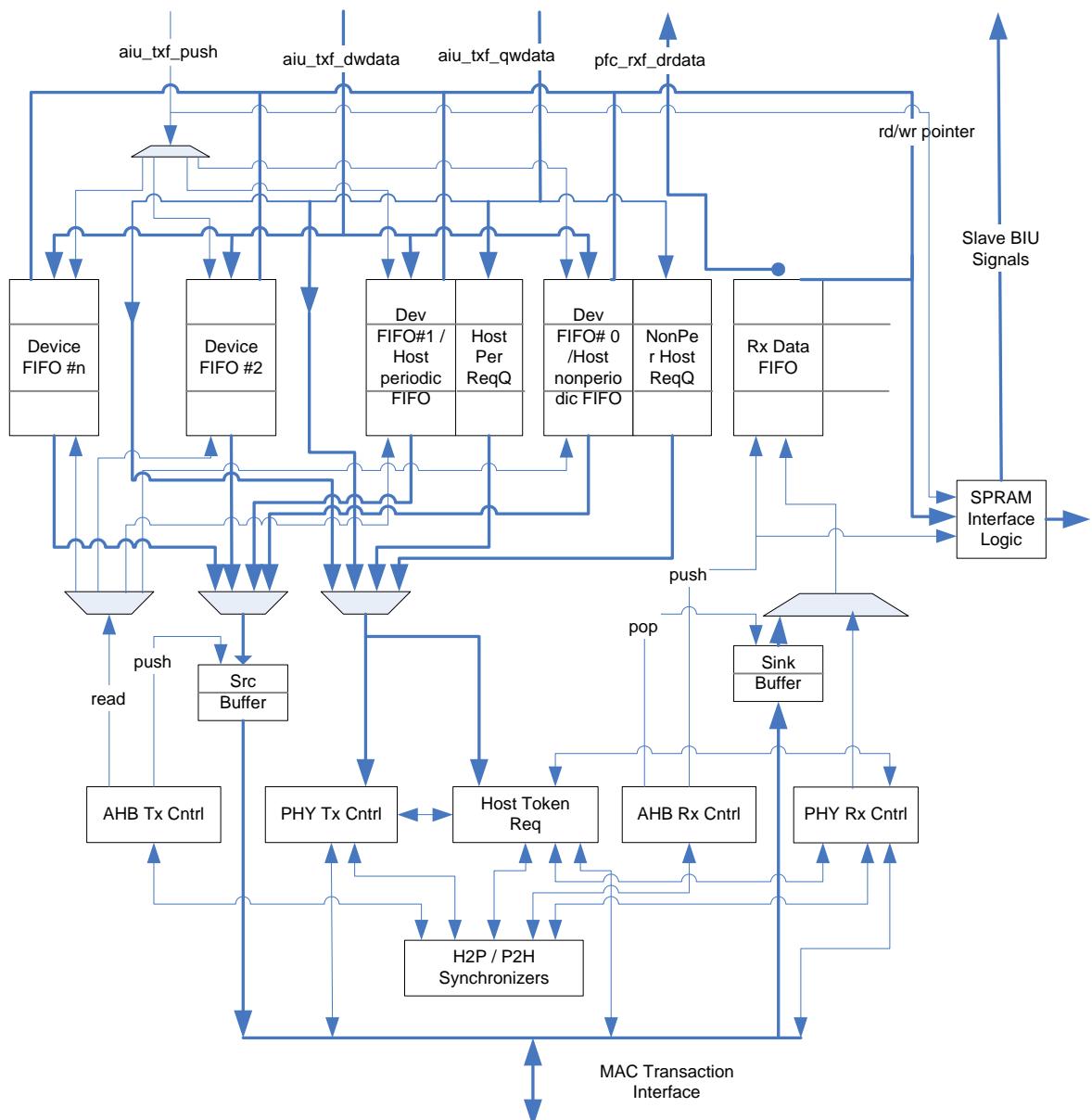


Figure 2-20 represents the Packet FIFO Controller in Shared FIFO operation.

**Figure 2-20 Packet FIFO Controller (Dedicated FIFO)**



Several FIFOs are used in Device and Host modes to store data inside the controller before transmitting it on either the AHB or the USB.

### 2.2.4.1 Periodic Transmit Data FIFO

This FIFO stores periodic transmit data. There are different periodic FIFO structures in Device and Host modes.

#### Device Mode

This topic includes shared FIFO operation and dedicated transmit FIFO operation.

##### <sup>1</sup>Shared FIFO operation

In Device mode, there is one FIFO per periodic endpoint, and the number of these FIFOs is determined during coreConsultant configuration. Each periodic FIFO is associated with a periodic IN endpoint using the DIEPCTLn.TxFNum field.

- The Slave BIU or the DMA Interface Unit writes data to this FIFO, and the MAC reads data from it.
- Each FIFO holds data for a single periodic IN endpoint. In a single microframe, this FIFO can only hold the data to be transmitted, which can be no more than three maximum packet size packets, in the case of high-bandwidth endpoints.
- This FIFO uses a store-and-forward mechanism, which means that all data to be transmitted in a single microframe must be written to the FIFO before any data can be read out by the MAC. Even when one of the three packets is already written to the FIFO, the MAC cannot read it until all three packets are written to the FIFO.
- If you selected Enable Dynamic FIFO Sizing? during coreConsultant configuration, the start address of the RAM allocated per FIFO must be programmed into the CSRs.
- The Periodic FIFO in Host mode is reused as the first periodic FIFO in Device mode.
- The Data FIFO is 35 bits wide:
  - Bits[34:32]: A 4-bit byte enable and a 1-bit packet delimiter are encoded into 3 bits and stored in external RAM. This reduces the data width in the external data RAM. This encoding is as follows:
    - 3'b100: Last DWORD of the packet, all 4 bytes are valid
    - 3'b011: Last DWORD of the packet, bytes 0, 1, and 2 are valid
    - 3'b010: Last DWORD of the packet, bytes 0, and 1 are valid
    - 3'b001: Last DWORD of the packet, only byte 0 is valid
    - 3'b111: Non-last DWORD of a packet, all 4 bytes are valid
  - Bits[31:0]: Data

1. This option is no longer supported, for more details, see [Note](#) on page 36.

## Dedicated Transmit FIFO Operation

In this state, there are no periodic or non-periodic FIFOs. FIFOs are implemented based on the number of IN endpoints (OTG\_NUM\_IN\_EPS) and the application chooses a specific FIFO for an IN endpoint when enabling an endpoint. The following periodic IN endpoint characteristics remain valid in Dedicated Transmit FIFO operation:

- The Slave BIU or the DMA Interface Unit writes data into a dedicated FIFO and the MAC reads the data from it.
- Each FIFO holds data for a single periodic IN endpoint. In the case of high-bandwidth endpoints, this FIFO can hold in a single microframe only the data to be transmitted, which can be no more than three maximum-packet-size packets.
- A whole packet must be in the FIFO before it can be transmitted on the USB when thresholding is not enabled.
- When thresholding is enabled, the controller starts transmitting on the USB in response to an IN token, when there is at least one MAC threshold amount of data available in the FIFO.

For high-bandwidth periodic IN endpoints, the controller can transmit one packet that is already in available in the FIFO, while a second packet is being pushed into the FIFO.

## Host Mode

This topic includes Shared FIFO Operation and Non-periodic Transmit Data FIFO operation.

### Shared FIFO Operation

In Host mode, a single periodic FIFO stores data for all periodic endpoints. A periodic Request queue associated with this FIFO stores periodic token requests.

- The Slave BIU or the DMA Interface Unit writes data into this FIFO and the MAC reads data out of this FIFO.
- This FIFO holds the data for all periodic channels.
- The depth of the Request queue is chosen during coreConsultant configuration, and this queue is implemented using flip-flops. This queue holds the channel numbers for both OUT and IN token requests.
- The Data FIFO stores data for OUT tokens. If you selected Enable Dynamic FIFO Sizing? during coreConsultant configuration, both the start address and the size of the Data FIFO must be programmed in the CSRs.
- The Data FIFO is 35 bits wide:
  - Bits[34:32]: A 4-bit byte enable and a 1-bit packet delimiter are encoded into 3 bits and stored in external RAM. This reduces the data width in the external data RAM. This encoding is as follows:
    - 3'b100: Last DWORD of the packet, all 4 bytes are valid
    - 3'b011: Last DWORD of the packet, bytes 0, 1, and 2 are valid
    - 3'b010: Last DWORD of the packet, bytes 0, and 1 are valid
    - 3'b001: Last DWORD of the packet, only byte 0 is valid
    - 3'b111: Non-last DWORD of a packet, all 4 bytes are valid
  - Bits[31:0]: Data

- The Request queue is 9 bits wide:
  - Bit [8]: Even/odd microframe to which the packet belongs
  - Bits [7:4]: Channel number
  - Bits [3:2]:
    - 2b'11: Disable channel
    - 2b'00: Zero-length packet
  - Others not used
- Bit [1]: Last periodic entry, used only in host Scatter/Gather DMA mode, else reserved.
- Bit [0]: Last request for the channel

#### 2.2.4.2 Non-Periodic Transmit Data FIFO

This FIFO stores the data to be transmitted on all non-periodic IN endpoints in Device mode and OUT channels in Host mode. A non-periodic Request queue associated with this FIFO stores endpoint- or channel-related information.

- The Slave BIU or the DMA Interface Unit writes data to this FIFO and the MAC reads this data from it.
- In Device mode, for each data packet in the Data FIFO there is a corresponding endpoint number and IN data packet type information entry in the Request queue.
- In Host mode, for each data packet in the Data FIFO there is a corresponding channel number and token-related control information entry in the Request queue.
- The Request queue also holds IN token entries in Host mode, even though there is no associated data in the Data FIFO.
- The depth of the Request queue is chosen during coreConsultant configuration and is implemented using flip-flops.
- If you selected Enable Dynamic FIFO Sizing? during coreConsultant configuration, the Data FIFO RAM size and start address must be programmed in the CSRs.
- The Data FIFO is 35 bits wide:
  - Bits [34:32]: A 4-bit byte enable and a 1-bit packet delimiter are encoded into 3 bits and stored in external RAM. This reduces the data width in the external data RAM.
  - This encoding is as follows:
    - 3'b100: Last DWORD of the packet, all 4 bytes are valid
    - 3'b011: Last DWORD of the packet, bytes 0, 1, and 2 are valid
    - 3'b010: Last DWORD of the packet, bytes 0, and 1 are valid
    - 3'b001: Last DWORD of the packet, only byte 0 is valid
    - 3'b111: Non-last DWORD of a packet, all 4 bytes are valid
  - Bits [31:0]: Data

- The Request queue is 7 bits wide:
  - Host Mode:
    - Bits [6:3]: Channel number
    - Bits [2:0]: Token-related control information
  - Device Mode:
    - Bits [6:3]: IN endpoint number
    - Bit [2]: Not used
    - Bit [1]: Indicates a zero-length data packet
    - Bit [0]: Not used

## Dedicated Transmit FIFO Operation

In Host mode, the FIFO architecture is the same as in non Dedicated Transmit FIFO Operation.

In Device mode, there are no periodic or non-periodic FIFOs. FIFOs are implemented based on the number of IN endpoints (OTG\_NUM\_IN\_EPS), and the application chooses a specific FIFO for an IN endpoint when it enables an endpoint. The following non-periodic IN endpoint characteristics remain valid in Dedicated Transmit FIFO mode:

- There are no queues associated with Device mode FIFOs.
  - If Dynamic FIFO Sizing is enabled, the data FIFO RAM size and the start address must be programmed in the CSRs
  - The data FIFO is 35 bits wide. Bits 31:0 are data, and bits 34:32 are byte enables and packet delimiter encoding. Bits 34:32 are decoded below:
    - 3'b100 - Last DWORD of the packet, all 4 bytes valid
    - 3'b011 - Last DWORD of the packet, Bytes 0, 1, 2 valid.
    - 3'b010 - Last DWORD of the packet, Bytes 0, 1 are valid.
    - 3'b001 - Last DWORD of the packet, only Byte 0 is valid.
    - 3'b111 - Non Last DWORD of a packet, all 4 bytes valid.
- 4-bit byte enables and 1-bit packet delimiter are encoded into 3 bits before storing into the external RAM. This reduces the data width in the external data RAM

## Receive Data FIFO

This FIFO stores data and status received on all OUT endpoints in Device mode and all IN endpoints in Host mode.

- The MAC writes data into this FIFO, and the Slave BIU or DMA Interface Unit reads this data from it.
- This FIFO can hold multiple OUT packets belonging to different endpoints or channels
- For each data OUT packet received on the USB, there is an associated status that indicates the byte count, packet status, and endpoint or channel number of the received packet.
- Data packets received with errors are flushed, so the Slave BIU and DMA Interface Unit never receive a corrupted data packet.

- SETUP data packets received by the MAC in Device mode are also written to the receive FIFO.
- Space is Reserved in the Receive Data FIFO to store SETUP packets and related status information, based on the control endpoint supported by the device. The controller never uses this Reserved space to store any other type of data. This ensures that there is always enough space in the receive Data FIFO to store SETUP data packets.
- If the device supports  $n$  endpoints,  $(4 * n + 6)$  DWORDs of space are allocated for receiving SETUP data packets. The extra 6 DWORDs are allocated to accommodate any back-to-back SETUP data packets sent by the USB host.
- The Data FIFO is 35 bits wide:
  - Bits [34:32]: Unused
  - Bits [31:0]: Data word
  - Device mode status:
    - Bits [24:21]: 4 LSBs of the frame number in which the packet was received
    - Bits [20:17]: Packet status
    - Bits [16:15]: Data PID
    - Bits [14:4]: Byte count
    - Bits [3:0]: Endpoint number
  - Host mode status:
    - Bits [20:17]: Packet status
    - Bits [16:15]: Data PID
    - Bits [14:4]: Byte count
    - Bits [3:0]: Channel number

#### 2.2.4.3 PFC-to-SPRAM Interface

Figure 2-21 shows how to connect the DWC\_otg Data FIFO interface to an industry-standard, single-port synchronous SRAM. Address, write data, and control outputs are driven late by the DWC\_otg, but in time to meet the SRAM setup requirements. Input read data is expected late from the SRAM and registered inside the controller before being used.

**Figure 2-21 DFIFO Single-Port Synchronous SRAM Interface**

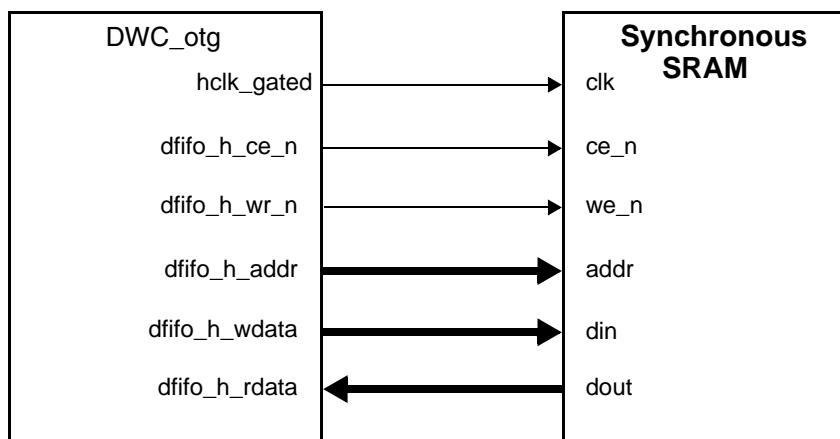
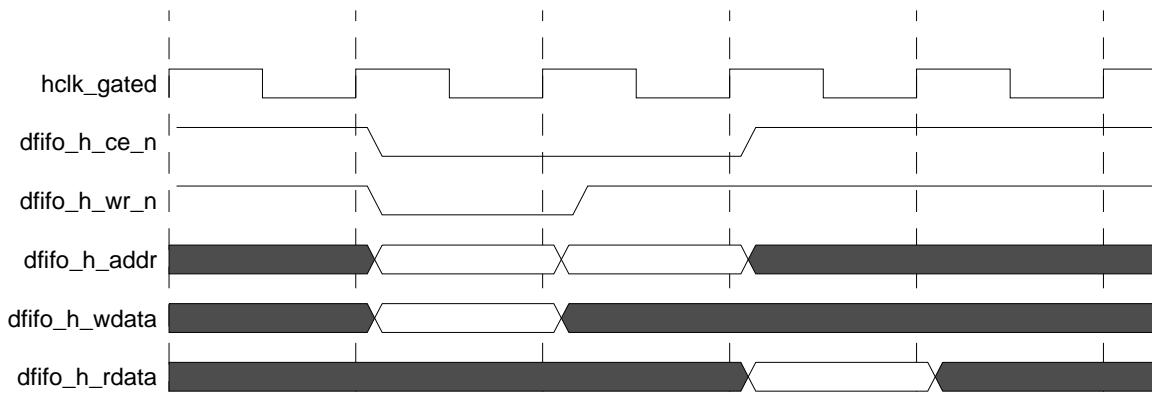


Figure 2-22 shows Data FIFO synchronous static RAM interface timing.

**Figure 2-22 Data FIFO Synchronous Static RAM Interface Timing**



#### 2.2.4.4 Endpoint Information Controller (EPINFO\_CTL)

The last locations in the SPRAM are used to hold register values. EPINFO\_CTL manages the stored values in the last few locations of the SPRAM as listed in Table 2-1.

**Table 2-1 SPRAM Register Values**

Mode	Feature Selected	SPRAM Space allocation (EP_LOC_CNT)
Host	Buffer DMA mode	One location per channel is used in SPRAM to store the HCDMAn value. For example, if there are ten channels, then the last ten SPRAM locations are reserved for storing the values.
Host	Scatter/Gather DMA mode	Four locations per channel are used in SPRAM to store the Base Descriptor address, Current Descriptor address, Current Buffer Pointer, and the Status Quadlet. For example, if there are ten channels, then the last forty locations are reserved for storing these values.
Device	Buffer DMA mode	One location per endpoint direction is used in SPRAM to store the DIEPDMA and DOEPDMA value. The application writes data and then reads it from the same location. For example, if there are ten bidirectional endpoints, then the last 20 SPRAM locations are reserved for storing the DMA address of IN and OUT endpoints.
Device	Scatter/Gather DMA mode	Four locations per endpoint direction are used in SPRAM to store the Base Descriptor address, Current Descriptor address, Current Buffer Pointer and the Status Quadlet. The application writes data to the base descriptor address. When the application reads the location where it wrote the base descriptor address, it receives the current descriptor address. For example, if there are ten bidirectional endpoints, then the last 80 locations are reserved for storing these values.

## 2.2.5 Media Access Controller

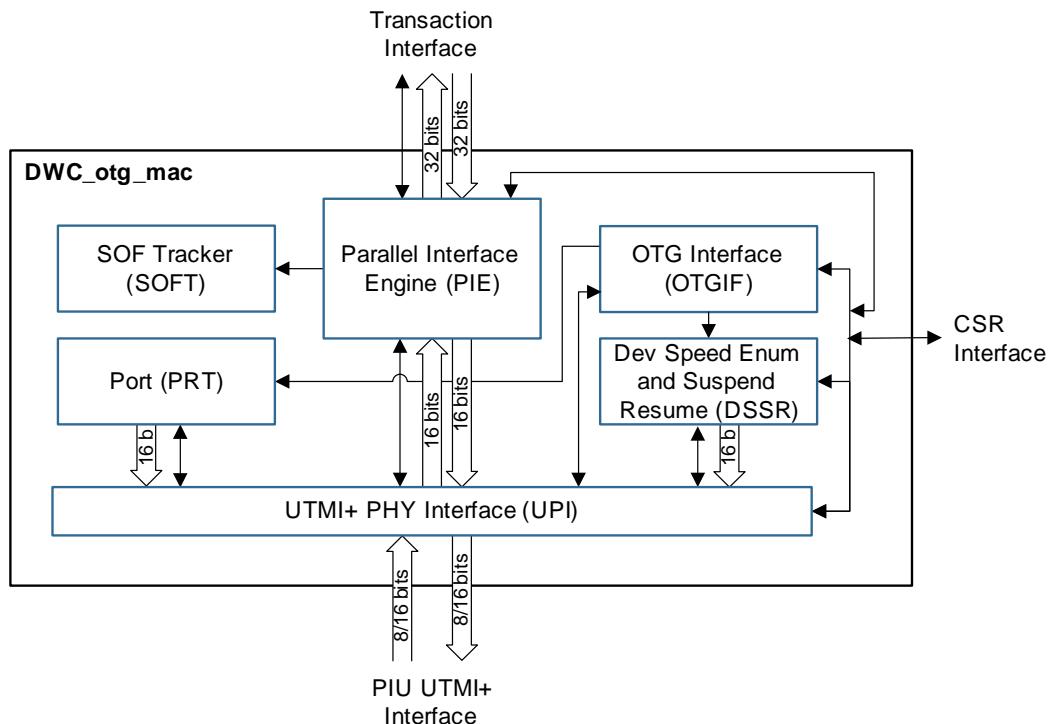
The Media Access Controller (MAC) module handles USB transactions, and device, host, and OTG protocols. This section discusses the following topics:

- ❑ “[MAC Components](#)” on page [92](#)
- ❑ “[USB Transaction Handling](#)” on page [93](#)
- ❑ “[Device Protocol Handling](#)” on page [94](#)
- ❑ “[Host Protocol Handling](#)” on page [94](#)
- ❑ “[OTG Protocol Handling](#)” on page [94](#)
- ❑ “[LPM Functions](#)” on page [94](#)

### 2.2.5.1 MAC Components

[Figure 2-23](#) shows the major MAC module components.

**Figure 2-23 MAC Block Diagram**



The major blocks are:

- Device Speed enumeration, Suspend, and Resume block (DSSR)

The DSSR block is only active in Device mode. This block performs the speed enumeration, suspend, resume and remote wakeup functions in Device mode.

- Parallel Interface Engine (PIE)

This block is responsible for token, data, and handshake packet generation and reception, and PID and CRC checking and generation. It generates handshake and data packets based on data integrity

and on CSR control and FIFO status information. The PIE also handles the data transfer to and from the FIFO, and the status update to the PFC and AIU.

- **SOF tracker (SOFT)**

This block tracks SOF packets and generates SOF interrupts in Device mode. It handles missing SOFs and delayed SOFs to keep the frame number synchronization between the host and the device.

- **Port (PRT)**

The Port block is only active in the Host mode. It is responsible for connect and disconnect detection, USB reset and speed enumeration, suspend and resume generation, remote wakeup detection, SOF generation, and High Speed Test mode handling.

- **OTG Interface (OTGIF)**

The OTG Interface block handles SRP and HNP. These OTG protocols are implemented either through the regular UTMI+ interface or the I<sup>2</sup>C interface (for USB 1.1 serial transceivers only).

- **UTMI+ PHY Interface (UPI)**

The block converts data widths for the 8-bit PHY interface and multiplexes output signals to the PHY from multiple blocks. It also implements some logic shared by multiple MAC blocks.

The MAC is designed so that unused components can be removed in some configurations to reduce gate count. SOFT, DSSR, and OTGIF can be removed in a host-only configuration; PRT and OTGIF can be removed for device-only configuration. All configuration activities are performed using coreConsultant. Refer to the *DesignWare Cores USB 2.0 Hi-Speed On-The-Go (OTG) QuickStart* for more information on coreConsultant.

### **2.2.5.2 USB Transaction Handling**

This topic discusses transactions in device and host modes.

#### **Device Mode**

In Device mode, the MAC decodes and checks the integrity of token packets as it receives them from the host. If the received token is a valid OUT or SETUP token, the MAC waits and checks the PID of the following data packet, then writes the data to the RxFIFO when it is available. After the packet is completed, the MAC checks the data integrity, sends the appropriate handshake when required to the host, and writes the transaction status to the receiving status queue. If the OUT token is received and the RxFIFO is not available, the MAC sends the host a NAK handshake. If the received token is a valid PING token, the MAC sends the appropriate handshake, based on the FIFO status and CSR control information. If an IN token is received and the data is available in the FIFO, the MAC reads the data, builds the data packet and sends it, waits for a handshake packet, if any, from the host, then updates the transaction status to the PFC. If an IN token is received and the data is not available in the FIFO, the MAC sends the host a NAK handshake.

#### **Host Mode**

In Host mode, the MAC receives a token request from the AIU to start a USB transaction. After receiving a token request, the MAC builds and sends the requested token packet. For OUT or SETUP transactions, the MAC reads data from the TxFIFO, builds and sends a data packet, waits for a handshake packet, if any, from the device, then updates the transaction status to the AIU. For an IN or PING transaction, the MAC waits for a data or handshake packet from the device. If it receives a handshake packet, the MAC updates

the status to the AIU. If it receives a data packet with the correct PID, the MAC writes the data into the RxFIFO, checks the data's integrity, sends a handshake packet, when required, to the device, then updates the status to the AIU.

#### 2.2.5.3 Device Protocol Handling

In Device mode, the MAC handles the USB reset sequence and speed enumeration process to determine the USB operating speed. The MAC detects USB suspend and resume signaling from the host, initiates remote wakeup, handles soft connect and disconnect, decodes and tracks SOF packets, and handles high-speed test modes.

#### 2.2.5.4 Host Protocol Handling

In Host mode, the MAC detects the device connect and disconnect, handles the USB reset and speed enumeration process, initiates USB suspend and resume, detects remote wakeup, generates SOF packets, and handles high-speed test modes.

#### 2.2.5.5 OTG Protocol Handling

The MAC handles Host Negotiation Protocol (HNP) for host and peripheral role swapping, and handles Session Request Protocol (SRP), which allows an A-device to turn off VBUS to save power when the USB bus is used, and provides a means for a B-device to request the A-device to activate VBUS.

#### 2.2.5.6 LPM Functions

In both device and host modes, a successful LPM transaction results in the controller going into L1 (sleep) state. The sleep state functions are performed by DSSR/PRT modules.

### Device Mode Functions

As a device, the HS OTG controller parses for LPM transactions, consisting of a token with EXT PID and an extended LPM token, and provides an appropriate response. The response can be an ACK, NYET, or STALL, or no response (ERROR response). The controller sends an ERROR response if any token is in error or if a timeout occurs between the EXT PID token and LPM token. The controller sends a STALL response if the received packet's requested link state is not L1 (Sleep). The controller sends a NYET response if any of the following conditions are met:

- NYET is the programmed application response.
- Any of the transmit FIFOs are not empty (default).
- Any of the non-ISOC transmit FIFOs are not empty, and the GLPMCFG[22] bit is set.
- The controller has an incomplete Control transfer, and the GLPMCFG[21] bit is set.

The controller sends an ACK response if ACK is the programmed application response and all transmit queues are empty. LPM token reception is indicated to the application by means of an interrupt, and the LPM token response, as well as the received parameters in the token, are updated in the GLPMCFG CSR register.

### Host Mode Functions

The application sends SET and TEST port control transfer type when the host port is connected to (root) hub. In this mode (Remote device mode), the response from hub reaches the application similar to any other control transfer.

When the host port is directly connected to device (Local Device mode), the application must program the GLPMCFG CSR register in the controller to send an LPM transaction. The controller checks for all the channels to be disabled and sends out the LPM transaction. If the response from the local device is an error, the host retries LPM transactions for the programmed number of times (GLPMCFG.RetryCnt) or until a valid response is received from the device. The device's response is indicated to the application by an interrupt and the LPM token response is updated in the CSR.

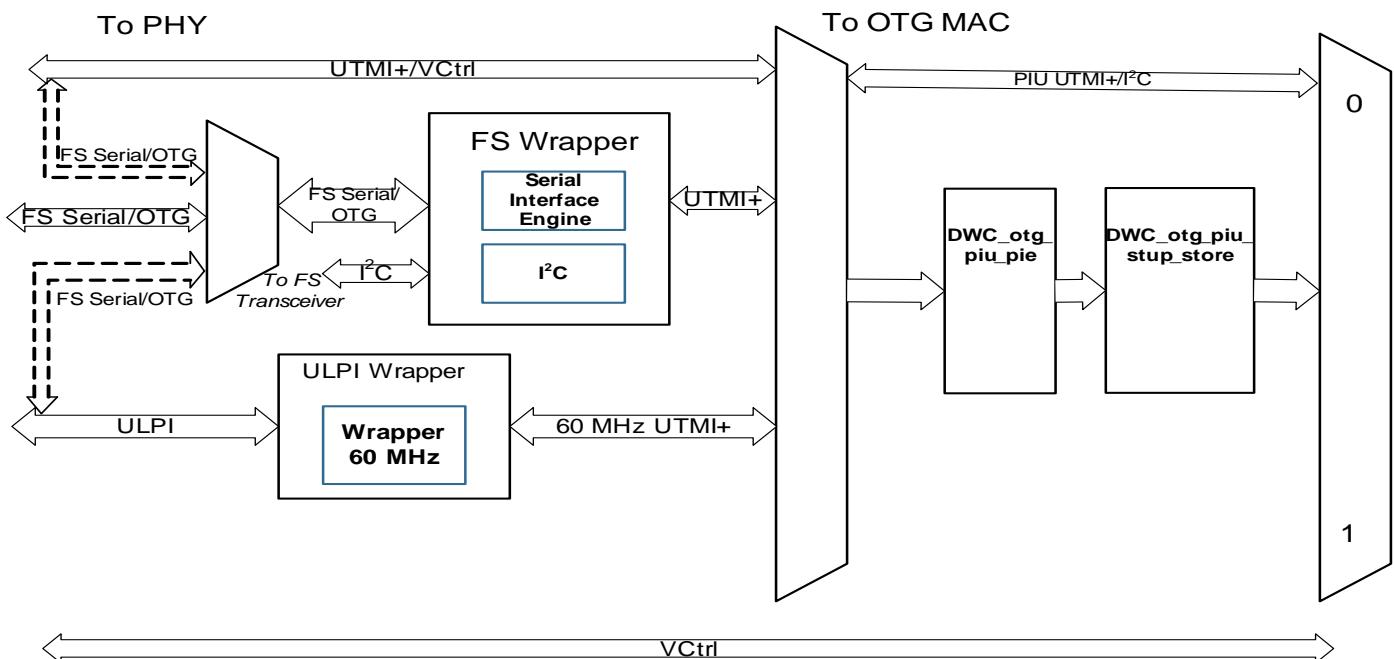
## 2.2.6 PHY Interface Unit (PIU)

This section explains the different PHY interfaces supported by the DWC\_otg controller. The PHY interface(s) available on the controller depends on the value specified for High-Speed PHY interface(s) and USB 1.1 Full-Speed Serial Transceiver interface during coreConsultant configuration. After power-on, you can select either the UTMI+, ULPI, or the USB 1.1 FS serial transceiver using software. The selection of Revision 1.0 USB or HSIC for UTMI interface and USB 1.1 FS or IC\_USB for serial transceiver is also software-/ strap pin-programmable.

PHYS and FS transceivers can be internal or external to the chip containing the DWC\_otg controller. Dedicated PHY/FS transceiver interfaces support both internal and external implementations. External USB 1.1 FS serial transceivers can share external UTMI+ or ULPI interface ports to reduce the number of pins on the chip.

[Figure 2-24](#) shows the PIU PHY and FS transceiver interfaces.

**Figure 2-24 PHY Interface Unit**



For more information on how to connect the PHY to the controller, see “Integrating with the PHY” in the [DesignWare Cores USB 2.0 Hi-Speed On-The-Go \(OTG\) User Guide](#).

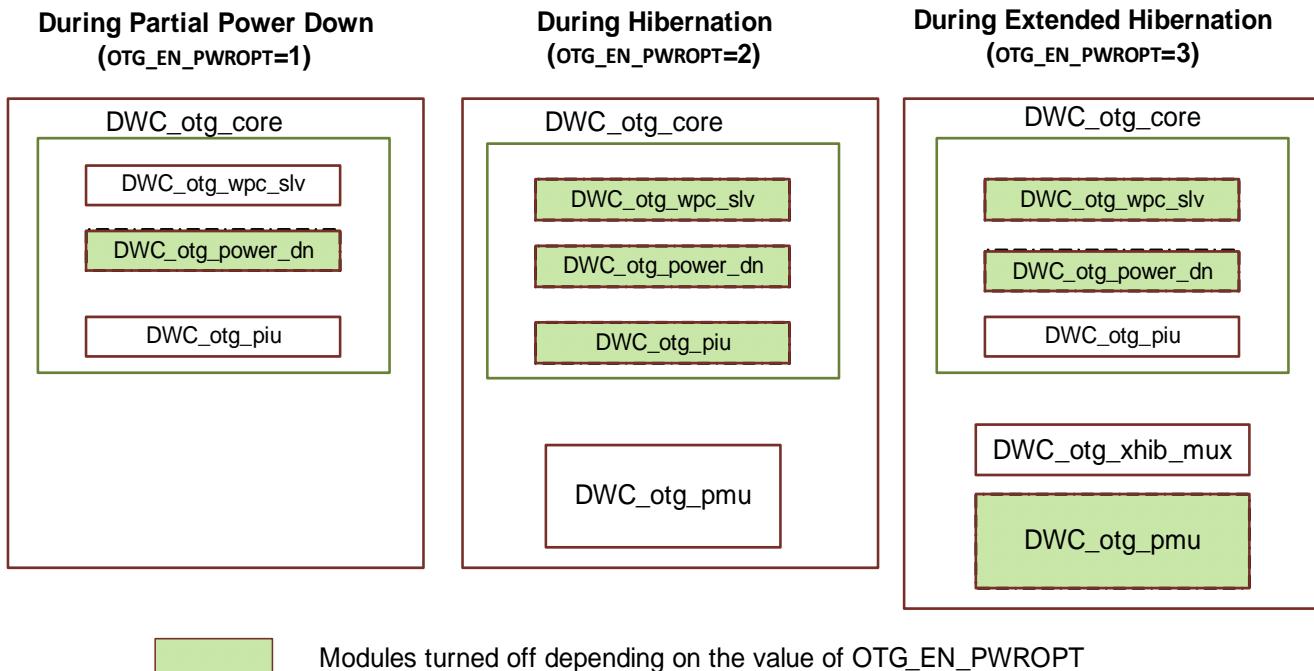
## 2.2.7 Wakeup and Power Controller (WPC)

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 DWC\_otg controller power consumption. To reduce power consumption further, the DWC\_otg controller also supports AHB clock gating and partial power-down. You can choose to turn off the power supply to some of the DWC\_otg modules or gate the AHB clock to some of the DWC\_otg modules when the USB is suspended or the session is not valid. See “FIFO RAM Allocation” in the Programming Guide for details on the programming sequences to use these power saving features. To use these features, you must set Enable Power Optimization during coreConsultant configuration.

When the USB is suspended or the session is not valid, the power supply to most DWC\_otg modules can be turned off. Some logic must remain powered on to detect the resume, remote wakeup, SRP or new session start event, then wake up the controller. To implement partial power-down, two power rails are needed. The logic that can be turned off in Partial Power-Down mode is on one power rail, VDD\_DN, and the logic that must be active in Partial Power-Down mode is on the other power rail, VDD\_WAKEUP.

The components involved in wake-up and power control are shown in [Figure 2-25](#). The BIU, PIU, WPC, CLK\_RST, and SYNC\_PWR blocks must be connected to the VDD\_WAKEUP domain. The CSR, AIU, PFC, MAC, and SYNC modules must be connected to the VDD\_DN domain. To avoid any power consumption in the powered-off modules, all signals generated by the powered-on modules and used by the powered-off modules are clamped low in Power-Down mode. To avoid state corruption from powered-off module signals, all signals generated by the powered-off modules and used by the powered-on modules are active high, and are clamped low when the power is off.

**Figure 2-25 Power Domains and Power Control Block Diagram**



The WPC module wakes up the controller from Partial Power Down mode. It has the following major functions.

**In Host mode:**

- Detects remote wakeup signaling from the device when the USB is suspended, drives resume, and deasserts the suspended pin
- Detects session requests (data line pulsing) from the device when the session is off, and deasserts the suspended pin
- Detects high resume signals to generate the PHY clock when the USB is suspended and the controller is in partial power-down mode
- Detects high resume signals to generate the PHY clock when the session is not valid and the controller is in partial power-down mode

**In Device mode:**

- Detects resume signaling from the host when the USB is suspended, and deasserts the suspended pin
- Detects SRP (data line pulsing) from the device when the USB is suspended, and deasserts the suspended pin
- Detects high resume signals to generate the PHY clock when the USB is suspended and the controller is in partial power-down mode
- Detects high resume signals to generate the PHY clock when the session is not valid and the controller is in partial power-down mode

### 2.2.8 List Processor

The List Processor (LP) implements Descriptor-Based Scatter/Gather DMA for Device and Host mode.

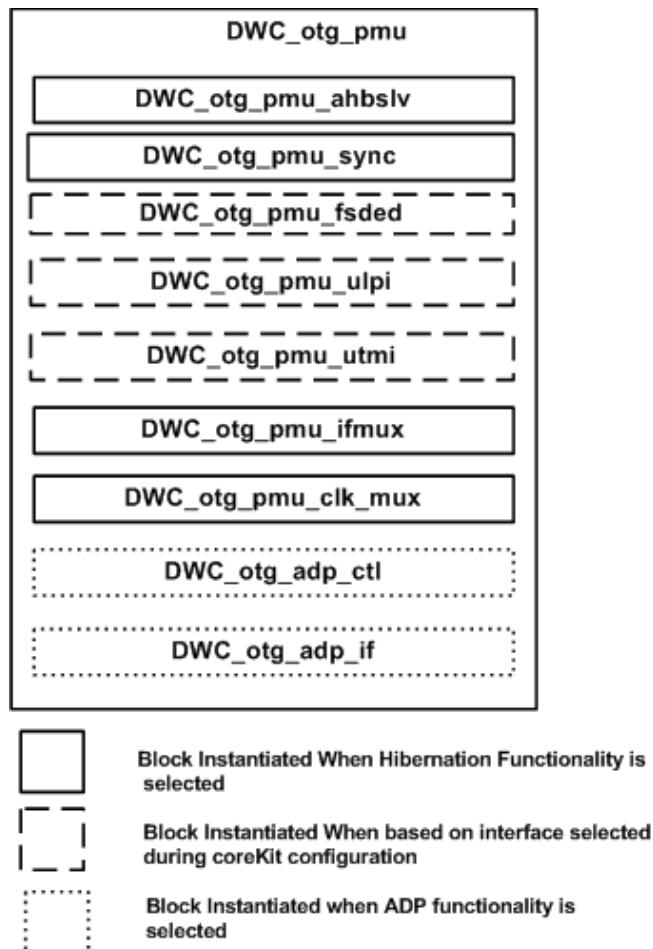


In this document, the terms Scatter/Gather DMA and Descriptor DMA are used interchangeably.

## 2.3 Power Management Unit Architecture

The Power Management Unit (PMU) is the module that supports hibernation feature. [Figure 2-26](#) illustrates the various blocks that constitute the DWC\_otg\_pmu module.

**Figure 2-26** [DWC\\_otg\\_pmu](#) Architecture



Following are the main components of the PMU module:

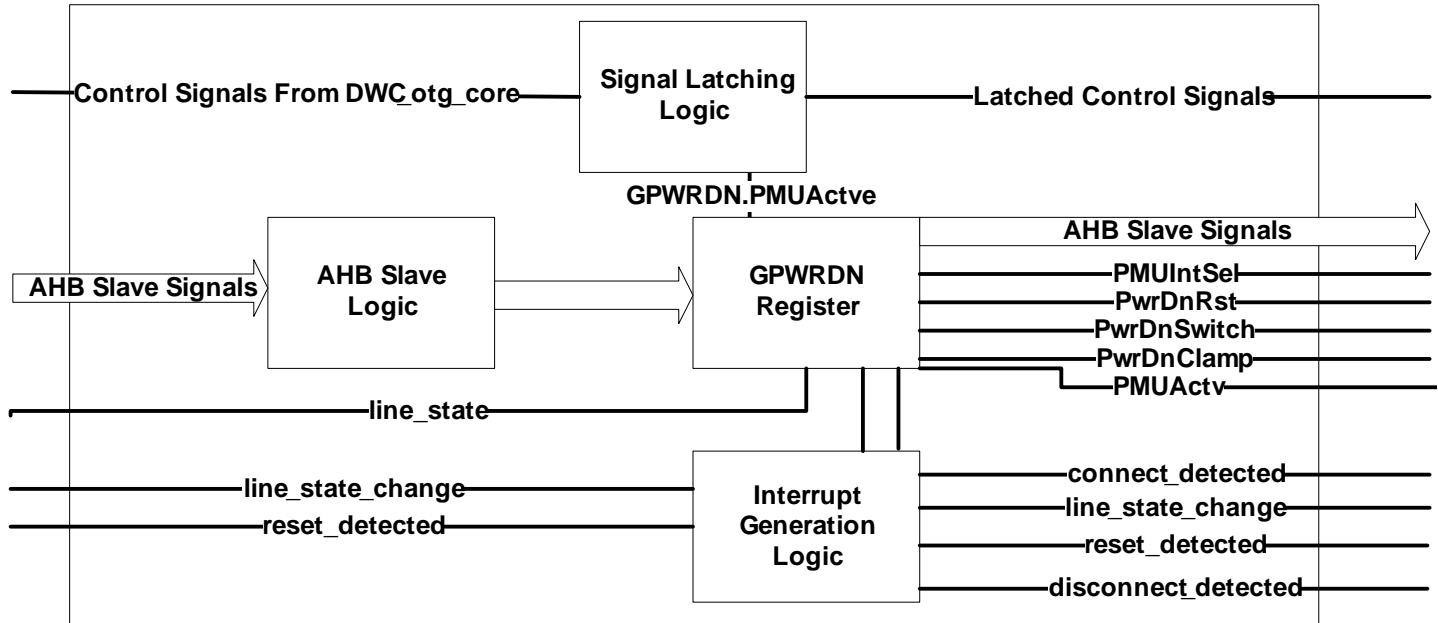
- “[PMU Slave](#)”
- “[PMU Logic](#)” on page [99](#)
- “[PMU IfMux Module](#)” on page [100](#)
- “[PMU Sync Module](#)” on page [101](#)
- “[ADP Module](#)” on page [102](#)

The following sections describe these components in detail.

### 2.3.1 PMU Slave

The PMU slave provides an application interface through the AHB bus to control the PMU logic. [Figure 2-27](#) illustrates the architecture of the PMU slave.

**Figure 2-27 Internal Architecture of PMU Slave**



AHB Slave Logic

This logic is used to write and read the GPWRDN register on the AHB slave interface.

Signal Latching Logic

This logic is used to latch the control signals from the DWC\_otg\_core before the controller is put into hibernation. The control signals are provided to the PMU logic to indicate the mode of operation of the controller.

GPWRDN register

This is the register implementation for the GPWRDN.

Interrupt Generation Logic

This logic is responsible for generation of the interrupt LineStateChange, ResetDetect, ConnectDetect, and DisconnectDetect to the application.

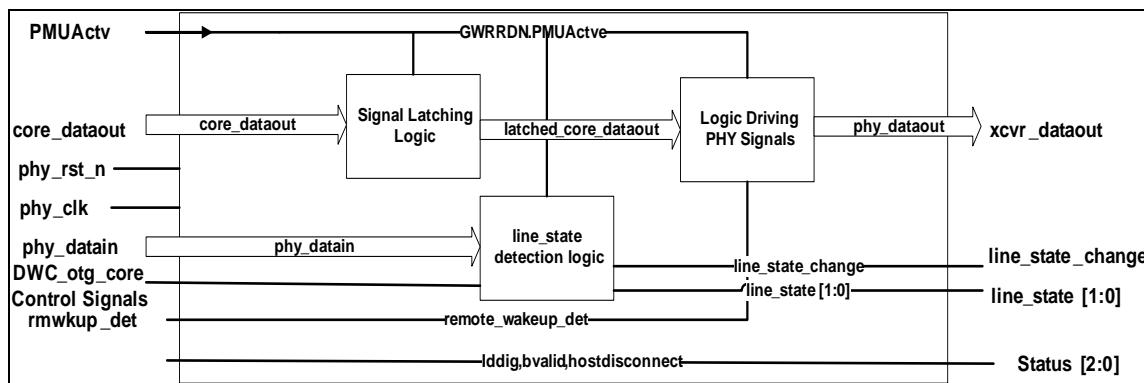
### 2.3.2 PMU Logic

Based on the PHY type being used, the PMU uses one of the following types of wake up logic:

- DWC\_otg\_pmu\_utmi - used for UTMI+ and HSIC interfaces
- DWC\_otg\_pmu\_fs\_ded - used for IC\_USB and FS Dedicated interfaces
- DWC\_otg\_pmu\_ulpi - used for ULPI interfaces

All the three types of PMU logic perform the following functions:

- Drives the PHY interface signals when the controller is in hibernation
- Decodes linestate for the DWC\_otg\_pmu\_ifmux module

**Figure 2-28 Internal Architecture of DWC\_otg\_pmu\_\*(utmi, ulpi, fs\_ded)**

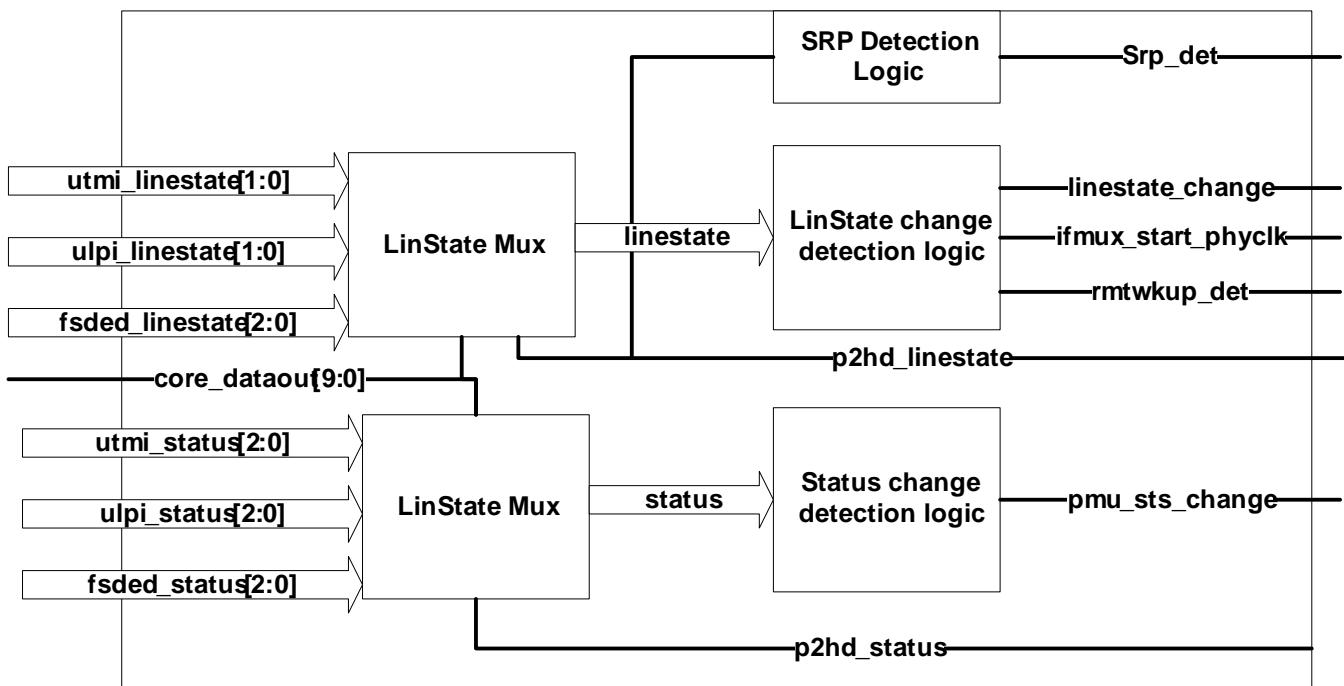
Apart from the PMU logic, the following logic is also used:

- |                                  |  |
|----------------------------------|--|
| <b>Signal Latching Logic</b>     | This logic is used to latch the controller output signals to the PHY when the wakeup logic is activated                          |
| <b>Logic Driving PHY Signals</b> | Based on normal operation, remote wakeup in host or device mode of operation, this logic drives the interface signals to the PHY |

### 2.3.3 PMU IfMux Module

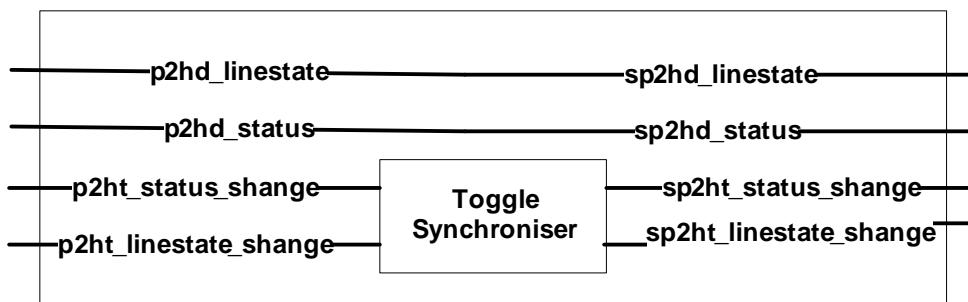
The DWC\_otg\_pmu\_ifmux module performs the following functions:

- Multiplex line\_state from different interfaces
- Detect changes in the line\_state
- Mutiplex status from different interfaces
- Detect changes in the status
- OTG Revision 2.0 and OTG Revision 1.3 SRP detection

**Figure 2-29 Internal Architecture DWC\_otg\_pmu\_ifmux**

### 2.3.4 PMU Sync Module

This module synchronizes the clock domain crossing signals between the DWC\_otg\_pmu\_ahb\_slv and the PMU interface logic. For more information on different types of synchronization, see “[Clock Domain Crossing](#)” on page [123](#).

**Figure 2-30 Internal Architecture of DWC\_otg\_pmu\_sync**

### 2.3.5 ADP Module

The ADP block within the PMU module supports ADP probing and ADP sensing activities according to the specifications in the *On-The-Go Supplement to the USB 2.0 Specification* (Revision 2.0). The ADP block consists of the following:

DWC\_otg\_pmu\_adp\_ctl.v This is the ADP control logic to PHY

DWC\_otg\_pmu\_adp\_ctl\_if.v This is the ADP control interface logic between AHB and PHY

DWC\_otg\_pmu\_sync.v This is the synchronizer between the AHB clock and the PHY clock



**Note** The ADP modules are present only if OTG\_ADP\_SUPPORT=1 while choosing the configuration.

---

## 2.4 System Clocks and Reset

The DWC\_otg controller has the following input clocks:

- hclk: AHB clock. hclk is the scan clock for the controller's AHB domain. Select it as a test clock.
- pmu\_hclk: Clock to PMU module when hibernation and/or ADP is enabled.



**Note** To use pmu\_hclk, you must check the clock insertion delay with the hclk, because you must balance the CTS between pmu\_hclk and hclk domain. You cannot use any other clock source for this input other than hclk.

- utmi\_clk: UTMI+ clock. Functionally used only when a UTMI PHY is selected, but always used as the PHY domain clock during DFT Scan mode. Select utmi\_clk as a test clock even when the controller is configured for a non-UTMI PHY.
- ulpi\_clk: ULPI PHY clock. Present only when a ULPI PHY is selected; negedge is also used in DDR mode. Select ulpi\_clk as a test clock when configuring the controller for a ULPI.
- utmifs\_clk48: Present only when OTG\_FSPHY\_INTERFACE != 0. The controller selects the utmi\_clk pin as the test clock when the controller is configured for USB 1.1 Full-Speed Serial Transceiver interface.
- ref\_clk: Present only when OTG\_SERV\_INT\_ENH is enabled. The controller selects utmi\_clk as a test clock for the logic sampled by ref\_clk.

In addition, the controller generates the following clocks in the DWC\_otg\_clkrst module:

- phy\_clk: Gated PHY clock output (one of the MUXed ULPI, UTMI+, USB 1.1 FS clocks, or the USB 1.1 FS clock divided by 8). When the USB 1.1 FS transceiver is selected, in LS the 48-MHz input clock is divided by 8 to generate a 6-MHz internal clock. This clock must operate in phase with, and have the same insertion delay as, the generated clock. If power optimization is enabled, phy\_clk must operate in phase with, and have the same insertion delay as, both the generated clock and wpc\_clk.
- wpc\_clk: Ungated PHY clock output (one of the MUXed ULPI, UTMI+, USB 1.1 FS clocks), which is present only when OTG\_EN\_PWR OPT is not equal to zero. This clock must operate in phase with, and have the same insertion delay as, phy\_clk, utmi\_clk, ulpi\_clk, and utmifs\_clk48.
- hclk\_gated: hclk gated for power optimization. Present only when OTG\_EN\_PWR OPT is not equal to zero.
- utmifs\_clk6: The ULPI, UTMI+, or USB 1.1 FS clock divided by 8. Present only when the USB 1.1 FS Serial Transceiver interface is selected.

## 2.4.1 System Clock Speeds

The selection of the system clock is summarized in [Table 2-2](#) and [Table 2-3](#).

**Table 2-2 System Clock Speeds**

Clock	Interface Selection	phy_clk
ulpi_clk: 60 MHz	<ul style="list-style-type: none"> <li>■ GUSBCFG.PHYSel = 1'b0</li> <li>■ GUSBCFG.ULPI_UTMI_Sel = 1'b1</li> <li>■ GUSBCFG.PHYIf = 1'b0</li> </ul> <p>High-speed ULPI PHY and MAC UTMI+ interfaces are 8 bits</p>	60 MHz (ulpi_clk)
utmifs_clk48: 48 MHz	<ul style="list-style-type: none"> <li>■ GUSBCFG.PHYSel = 1'b1</li> <li>■ GUSBCFG.ULPI_UTMI_Sel = 1'bx</li> <li>■ GUSBCFG.PHYIf = 1'bx</li> <li>■ OTG_FSPHY_INTERFACE = 3</li> </ul> <p>Full-speed PHY Interface: FS pins shared with ULPI</p>	FS: 48 MHz (utmifs_clk48) LS: 6 MHz (utmifs_clk48/8) (FS or LS is controlled by internal wpc_xcvrselect signal.)
utmi_clk: 30 MHz	<ul style="list-style-type: none"> <li>■ GUSBCFG.PhLPwrClkSel = 1'b0</li> <li>■ GUSBCFG.PHYSel = 1'b0</li> <li>■ GUSBCFG.ULPI_UTMI_Sel = 1'b0</li> <li>■ GUSBCFG.PHYIf = 1'b1</li> </ul> <p>High-speed UTMI+ PHY and MAC UTMI interfaces are 16 bits</p>	30 MHz (utmi_clk)
utmi_clk: 60 MHz	<ul style="list-style-type: none"> <li>■ GUSBCFG.PhLPwrClkSel = 1'b0</li> <li>■ GUSBCFG.PHYSel = 1'b0</li> <li>■ GUSBCFG.ULPI_UTMI_Sel = 1'b0</li> <li>■ GUSBCFG.PHYIf = 1'b1</li> </ul> <p>High-speed UTMI+ PHY and MAC UTMI+ interfaces are 8 bits</p>	60 MHz (utmi_clk)
utmifs_clk48: 48 MHz	<ul style="list-style-type: none"> <li>■ GUSBCFG.PhLPwrClkSel = 1'b1</li> <li>■ GUSBCFG.PHYSel = 1'b0</li> <li>■ GUSBCFG.ULPI_UTMI_Sel = 1'b0</li> <li>■ GUSBCFG.PHYIf = 1'b1</li> </ul> <p>For the HS UTMI+ PHY interface in Low Power mode, the PHY uses a 48 MHz clock for reference in FS/LS mode with the MAC UTMI+ 8-bit interface, instead of the 480 MHz clock.</p>	FS: 48 MHz (utmifs_clk48) LS: 48 or 6 MHz (PHY vendor dependent. Synopsys UTMI+ PHY provides 48 MHz in LS mode.)
utmifs_clk48: 48 MHz	<ul style="list-style-type: none"> <li>■ GUSBCFG.PHYSel = 1'b1</li> <li>■ GUSBCFG.ULPI_UTMI_Sel = 1'bx</li> <li>■ GUSBCFG.PHYIf = 1'b0</li> <li>■ OTG_FSPHY_INTERFACE = 2</li> </ul> <p>Full-speed PHY interface: FS pins shared with UTMI+</p>	FS: 48 MHz (utmifs_clk48) LS: 6 MHz (utmifs_clk48/8) (FS or LS is controlled by internal wpc_xcvrselect signal.)

**Table 2-2 System Clock Speeds (Continued)**

Clock	Interface Selection	phy_clk
utmifs_clk48: 48 MHz	<ul style="list-style-type: none"> <li>■ GUSBCFG.PHYSel = 1'b1</li> <li>■ GUSBCFG.ULPI_UTMI_Sel = 1'bx</li> <li>■ GUSBCFG.PHYIf = 1'bx</li> <li>■ OTG_FSPHY_INTERFACE = 1</li> </ul> <p>USB 1.1 Full Speed Serial Transceiver interface: Dedicated FS interface</p>	FS: 48 MHz (utmifs_clk48) LS: 6 MHz (utmifs_clk48/8) (FS or LS is controlled by internal wpc_xcvrselect signal.)

**Table 2-3 Tested Frequencies for Clocks**

Clock	Programming Selection	Tested Frequencies
ref_clk	GREFCLK.REFCLKPER = ref_clk period in ps	12 MHz, 16 MHz, 17 MHz, 19.2 MHz, 20 MHz, 24 MHz, 30 MHz, 40 MHz
hclk	-	50 MHz, 52 MHz, 60 MHz, 100 MHz, 150 MHz, 166 MHz

Depending on whether hibernation is enabled or not, there are some differences in functionality.

#### 2.4.1.1 Impact of System Clocks on Minimum Inter-Packet Gap and Number of Cascaded Hubs Supported by the Device

When the DWC\_otg controller acts as a device, there is a worst-case response time for a receive followed by a receive, as per the *UTMI Specification*. This worst-case response time depends on the AHB clock frequency.

The controller registers are in the AHB domain, and the controller does not accept another token before updating these register values. This worst-case value is seven PHY clocks when the AHB clock is the same as the PHY clock. When AHB clock is faster, this value is smaller.

The following scenarios are for a receive followed by a receive:

- Any response token from host (ACK/NAK/NYET/STALL) followed by a token
- As a special case, ISOC OUT Data packet followed by the next token

##### 2.4.1.1.1 System-Level Impact

At the system level, this limitation can reduce the performance of the device controller because the controller drops or sends NAK to the token which is sent by the host too close to end of the preceding ISOC OUT token. For bulk, interrupt, and control transactions, the system recovers as the host retires the token, and the transactions will eventually complete. For isochronous transactions, the host proceeds to the next transaction scheduled for the next bInterval.

A significant performance degradation occurs in the following cases:

- The device is connected to a host which is capable of consistently sending a token within 88-96 bit times, that is, six to seven PHY clocks from the end of the preceding ISOC OUT token.
- The device is connected to a host through multiple layers of hubs such that the inter-packet gap between an ISOC OUT token and the following token shrinks to less than seven PHY clocks.

However, in neither of these cases, the system stops responding nor enters an unrecoverable state.

#### 2.4.1.1.2 Internal Operation of DWC\_otg Controller in This Scenario

[Figure 2-31](#) on page 106 depicts the scenario of a token being received after an ISOC OUT token at the UTMI interface of the device controller. The markers in [Figure 2-31](#) on page 106 show the internal key events happening in the controller. [Table 2-4](#) on page 107 provides the description of these markers.

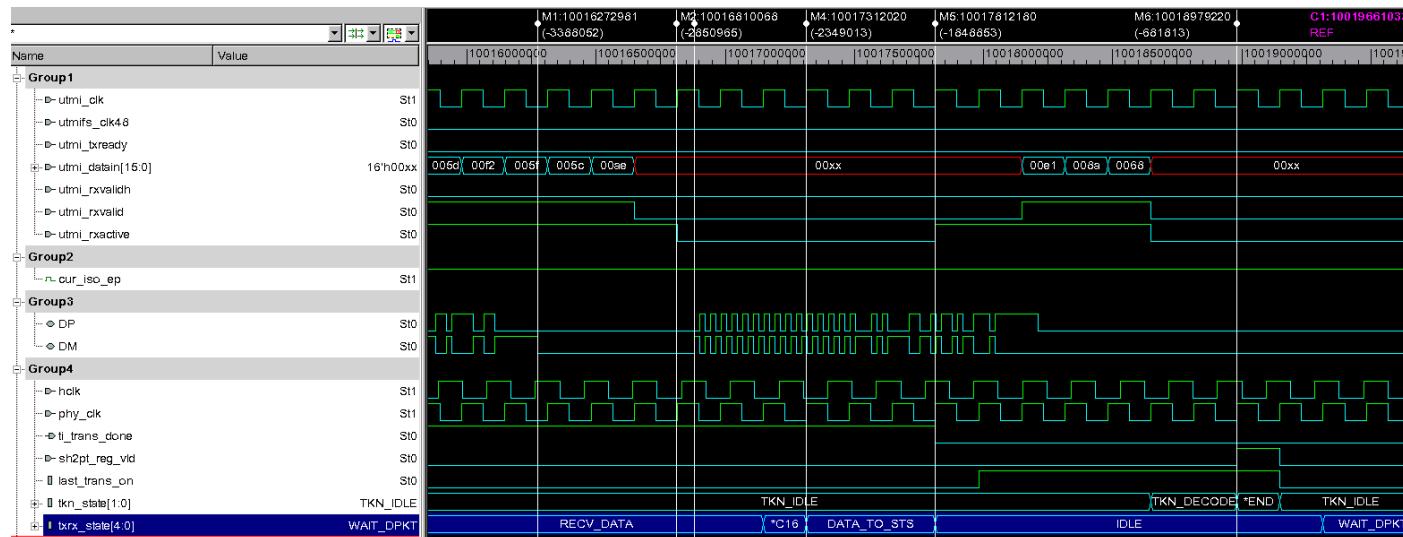
The following is the sequence of the controller internal events:

1. Process the received ISOC OUT token in the PHY clock (PCLK) domain.
2. Update the relevant registers in the AHB clock (HCLK) domain.
3. Wait for a completion handshake from the AHB clock domain to the PHY clock domain before processing the next token received on the UTMI interface.

The following are the internal signals involved:

- The **ti\_trans\_done** signal is a toggle signal generated in the PHY clock domain to update the AHB clock domain registers. The controller waits for a handshake from the AHB clock domain for this register update to accept the next token.
- The **sh2pt\_reg\_vld** signal is the handshake signal from the AHB clock domain indicating that the register update is complete in the AHB clock domain.
- The **txrx\_state** signal is in RCVD\_DATA state while pushing the received data to the RxFIFO, CHKCRC16 state while computing CRC, and DATA\_TO\_STS state while pushing the received status for the packet.
- The **tkn\_state** signal is in TKN\_DECODE state while decoding the received token.

**Figure 2-31 Controller Signals for ISOC OUT Token Followed by Next Token**



**Table 2-4 Description of Waveform Markers**

<b>Marker Reference</b>	<b>Description of Time Between Markers</b>
M5-M3	RX start delay Time between the first bit of the SYNC pattern on the DP/DM lines and the RXACTIVE assertion
M2-M1	RX end delay Time between the last bit of the EOP and the RXACTIVE de-assertion
M3-M1	RX to RX inter-packet delay measured at the DP/DM lines
M4-M2	For 16-bit UTMI mode: One PHY clock period from the rx_active de-assertion to the CRC calculation For 8-bit UTMI mode: Three PHY clock periods from the rx_active de-assertion to the CRC calculation Time between the RXACTIVE de-assertion to txrx_state=CHK_CRC16
M5-M4	Three PHY clock period for Receive packet status push Time taken by the txrx_state signal in the DATA_TO_STS state
M6-M5	Register update toggle pulse synchronization delay + Register update toggle pulse handshake synchronization delay Time between the ti_trans_done signal toggle to the sh2pt_reg_vld pulse
M6-M5	Seven PHY clock periods to receive and decode the token at PIE Time between the RXACTIVE assertion and tkn_state=TKN_END

## Calculation of Minimum AHB Clock Frequency Required

For calculating the minimum AHB clock frequency required to overcome the ISOC OUT inter-packet gap limitation, the time needed by the controller to complete the following two processes needs to be determined:

- Update the controller registers in AHB clock domain after receiving the ISOC OUT token
- Decode the next received token

To overcome this limitation so that the controller does not drop any token received too close after an ISOC OUT token, the time needed to update the controller registers in AHB clock domain should be less than or equal to the time needed to decode the next received token. This condition forms the basis for calculating the minimum AHB clock frequency needed.

### *Time Needed to Update the Controller Registers in AHB Clock Domain After rx\_active De-assertion*

The time needed to update the controller registers in the AHB clock domain after the rx\_active de-assertion

- = 1 PHY clock period from the rx\_active de-assertion to the CRC calculation in a 16-bit UTMI mode, or 3 PHY clock periods from the rx\_active de-assertion to the CRC calculation in a 8-bit UTMI mode
- + 3 PHY clock periods for Receive packet status push
- + 1.5 HCLK periods CDC max delay for the register update toggle pulse.
- + 2 to 3 HCLK periods for the register update toggle pulse synchronization
- + 1.5 PHY clock periods CDC max delay for the register update toggle handshake pulse
- + 2 to 3 PHY clock period for the register update toggle handshake pulse synchronization

In 16-bit UTMI mode, the time needed to update the controller registers in the AHB clock domain after the rx\_active de-assertion

- = 7.5 PHY clock period + 3.5 HCLK periods for the best-case synchronization delay
- or,
- = 8.5 PHY clock period + 4.5 HCLK periods for the worst-case synchronization delay

In 8-bit UTMI mode, the time needed to update the controller registers in the AHB clock domain after the rx\_active de-assertion

- = 9.5 PHY clock period + 3.5 HCLK periods for best case synchronization delay
- or,
- = 10.5 PHY clock period + 4.5 HCLK periods for worst case synchronization delay

### *Time Needed to Decode the Next Received Token*

In 16-bit UTMI mode, the time needed to decode the next received token after the rx\_active de-assertion

$$\begin{aligned}
 &= - 28 \text{ HS bit times of RX End delay} \\
 &\quad + \text{RX to RX inter-packet delay} \\
 &\quad + 48 \text{ HS bit times of RX start delay} \\
 &\quad + 3 \text{ PHY clock periods to receive and decode token at PIE} \\
 &= 141.59\text{ns} + \text{RX to RX inter-packet delay}
 \end{aligned}$$

In 8-bit UTMI mode, the time needed to decode the next received token after the rx\_active de-assertion

$$\begin{aligned}
 &= - 26 \text{ HS bit times of RX End delay} \\
 &\quad + \text{RX to RX inter-packet delay} \\
 &\quad + 45 \text{ HS bit times of RX start delay} \\
 &\quad + 7 \text{ PHY clock periods to receive and decode token at PIE} \\
 &= 156.14\text{ns} + \text{RX to RX inter-packet delay}
 \end{aligned}$$

### *AHB Clock Frequency Requirement*

This section discusses the AHB clock frequency requirements.

For 16-bit UTMI PHY and best-case synchronization delay:

$$\begin{aligned}
 7.5 \text{ PHY clock period (250.26 ns)} + 3.5 \text{ HCLK period} &= 141.70 \text{ ns} + \text{RX to RX inter-packet delay} \\
 \text{HCLK period for 88-bit time inter-packet delay} &= 21.28 \text{ ns, that is, } 46.99 \text{ MHz} \\
 \text{Minimum inter-packet delay supported with 53-MHz HCLK frequency} &= 83.93 \text{ HS bit times} \\
 \text{Minimum inter-packet delay supported with 250-MHz HCLK frequency} &= 58.92 \text{ HS bit times}
 \end{aligned}$$

For 16-bit UTMI PHY and worst-case synchronization delay:

$$\begin{aligned}
 8.5 \text{ PHY clock period (283.62 ns)} + 4.5 \text{ HCLK period} &= 141.70 \text{ ns} + \text{RX to RX inter-packet delay} \\
 \text{HCLK period for 88-bit time inter-packet delay} &= 9.13 \text{ ns, that is, } 109.44 \text{ MHz} \\
 \text{Minimum inter-packet delay supported with 53-MHz HCLK frequency} &= 109.05 \text{ HS bit times} \\
 \text{Minimum inter-packet delay supported with 250-MHz HCLK frequency} &= 76.88 \text{ HS bit times}
 \end{aligned}$$

For 8-bit UTMI PHY and best-case synchronization delay:

$$\begin{aligned}
 9.5 \text{ PHY clock period (158.49 ns)} + 3.5 \text{ HCLK period} &= 156.30 \text{ ns} + \text{RX to RX inter-packet delay} \\
 \text{HCLK period for 88-bit time inter-packet delay} &= 51.67 \text{ ns, that is, } 19.35 \text{ MHz (Min. stipulated 30 MHz)} \\
 \text{HCLK period for 32-bit time inter-packet delay} &= 18.39\text{ns, that is, } 54.37 \text{ MHz} \\
 \text{Minimum inter-packet delay supported with 53-MHz HCLK frequency} &= 32.80 \text{ HS bit times}
 \end{aligned}$$

For 8-bit UTMI PHY and worst-case synchronization delay:

$$10.5 \text{ PHY clock period (175.18 ns)} + 4.5 \text{ HCLK period} = 156.30 \text{ ns} + \text{RX to RX inter-packet delay}$$

HCLK period for 88-bit time inter-packet delay = 36.48 ns, that is, 27.41MHz (Min. stipulated 30 MHz)

HCLK period for 32-bit time inter-packet delay = 10.59 ns, that is, 94.36 MHz

Minimum inter-packet delay supported with 53-MHz HCLK frequency = 49.89 HS bit times

### Summary of Minimum AHB Clock Frequency Required Vs. Minimum Inter-Packet Gap

For a given UTMI PHY data width, Table 2-5 lists the minimum AHB clock frequency required for the controller to operate without any packet drop with the listed minimum inter-packet gap.

**Table 2-5 Minimum AHB Clock (HCLK) Frequency Required Vs. Minimum Inter-Packet Gap**

Minimum HCLK Frequency Required (in MHz)	Minimum Inter-Packet Gap (in HS Bit Times)
<i>UTMI PHY Data Width = 16</i>	
53	109.05
109.44	88
250	76.88
<i>UTMI PHY Data Width = 8</i>	
53	49.89
27.41	88
<b>94.36 (96 MHz recommended)</b>	<b>32</b>

In 16-bit UTMI mode,

- The controller can operate without any packet drop with 110 HS bit times minimum inter-packet gap for 53-MHz HCLK.
- The controller does not support 32 HS bit times minimum inter-packet gap.

In 8-bit UTMI mode,

- The controller can operate without any packet drop with 50 HS bit times minimum inter-packet gap for 53-MHz HCLK.
- The controller supports 32 HS bit times minimum inter-packet gap with 96-MHz HCLK.



- With 96-MHz HCLK and 16-bit UTMI, the controller can support 91 HS bit times inter-packet gap. If possible, use 110-MHz HCLK so that 88 HS bit times is met for the 16-bit UTMI.
- From 4.10a release, the scenario of ISOC OUT Data packet followed by any token is fixed. However, the non-ISOC scenarios still have the limitation (For details, see "[Worst-Case Inter-Packet Gap and Maximum Number of Cascaded Hubs Supported by Device Controller](#)" on page [49](#)).

## Impact of System Clocks on Number of Cascaded Hubs Supported by the Device

For details, see "[Effect of System Clocks on Number of Cascaded Hubs Supported by the Device](#)" on page [57](#).

## 2.4.2 System Clock and Reset Generation

### 2.4.2.1 AHB Clock Frequency and AHB Latency Requirements

In the presence of ISOC IN endpoints in the configuration in Device mode:

- The AHB Master should be given sufficient priority in the AHB bus such that it can fetch the complete (scheduled) packet before the start of scheduled microframe.
- Even in the worst case, the AHB Master should not take more than one microframe to fetch a single scheduled packet.

In Device mode, the AHB frequency must be selected so that both the controller and the PHY meet the Inter-Packet Delay requirement in section 6.4 of UTMI specification. In the case of IN tokens, make sure that the sum of RX End Delay (of the PHY) + GUSBCFG.USBTrdTim + TX Start Delay (of the PHY) is within 192bitTimes.

[Figure 2-32](#) shows the implementation of a logic for system clocks and the reset generation.

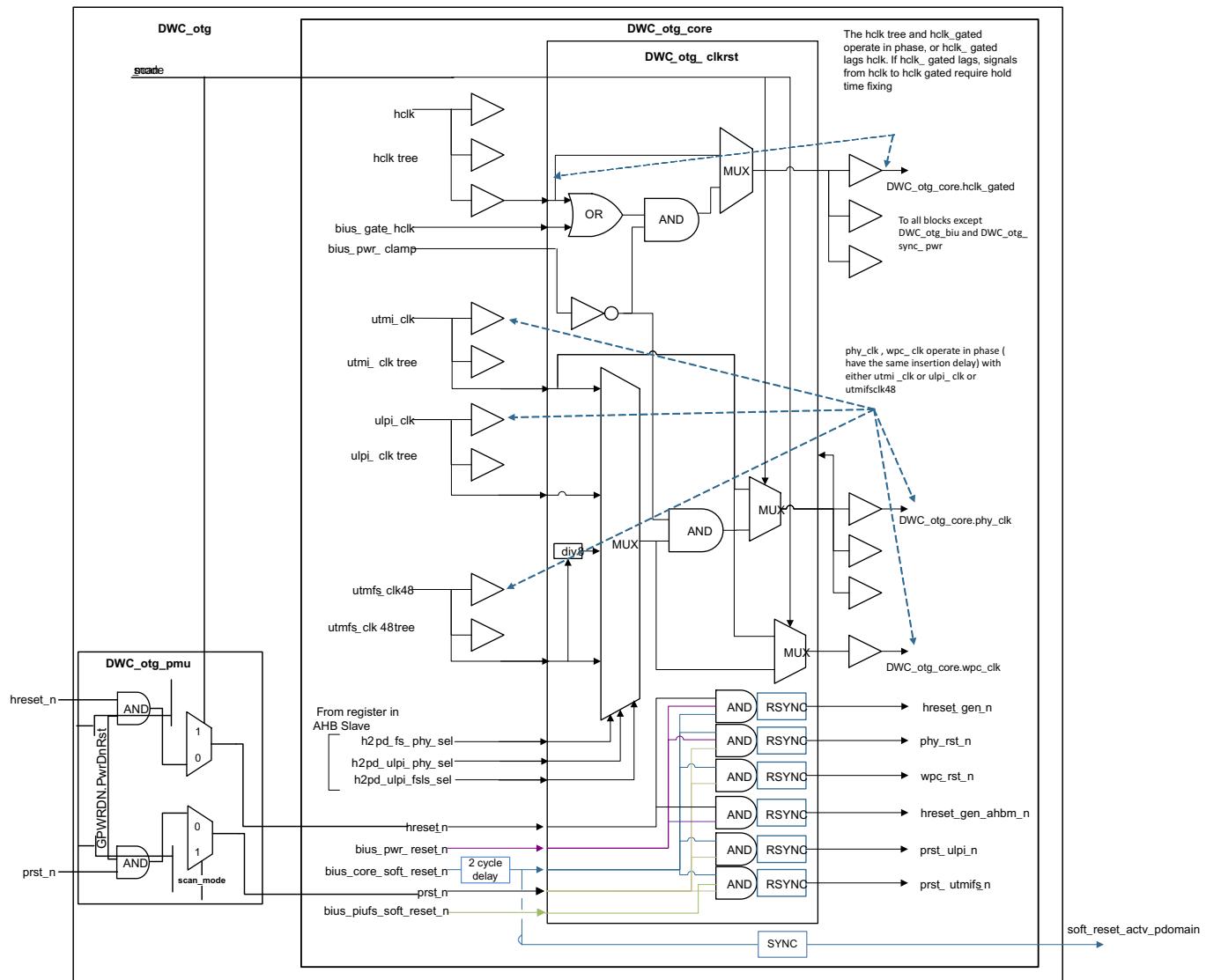
**Figure 2-32 System Clock and Reset Generation**

Figure 2-33 shows the additional logic for system clocks and reset generation when PMU is instantiated.

**Figure 2-33 Additional Logic for System Clocks and Resets When PMU is Instantiated**

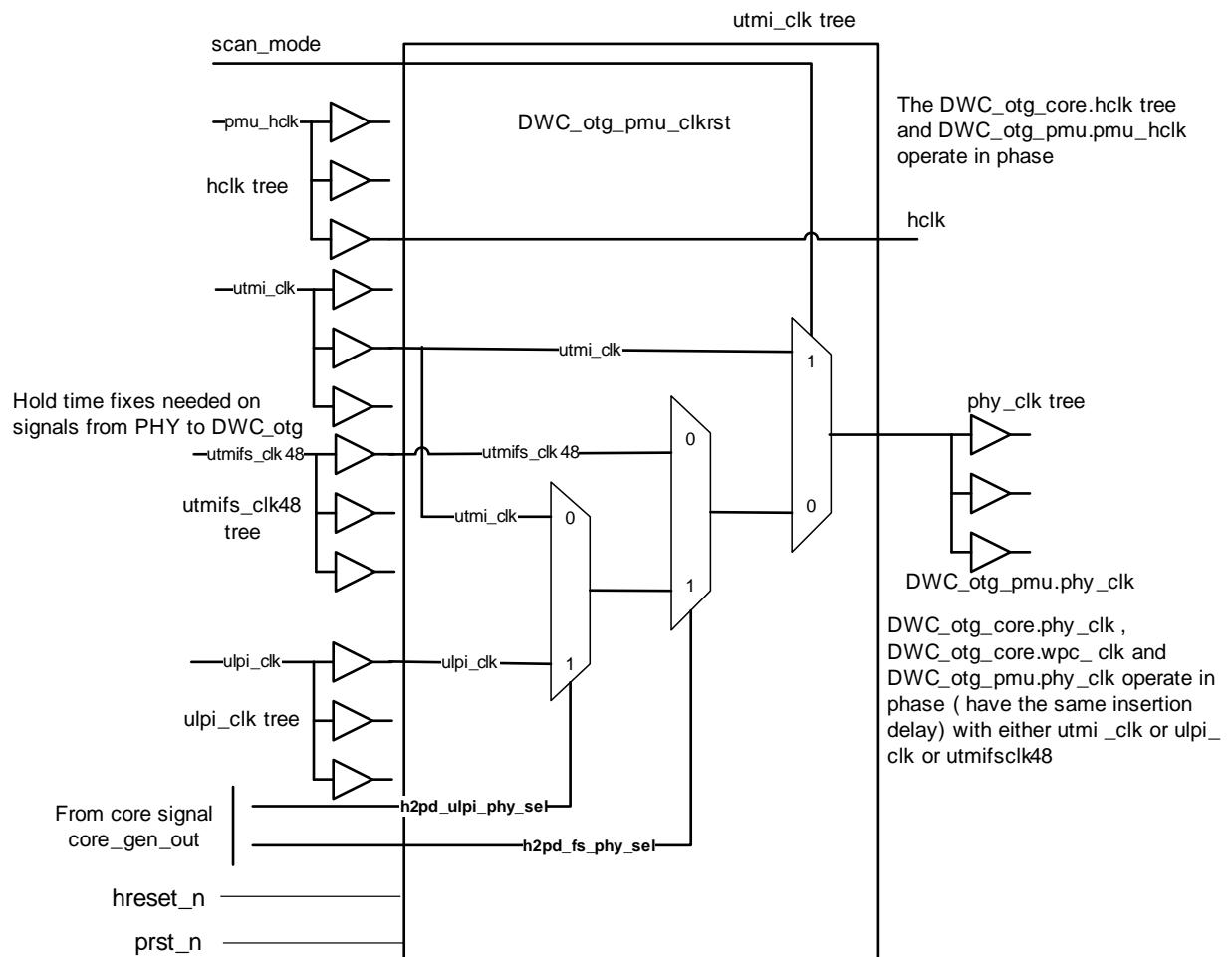
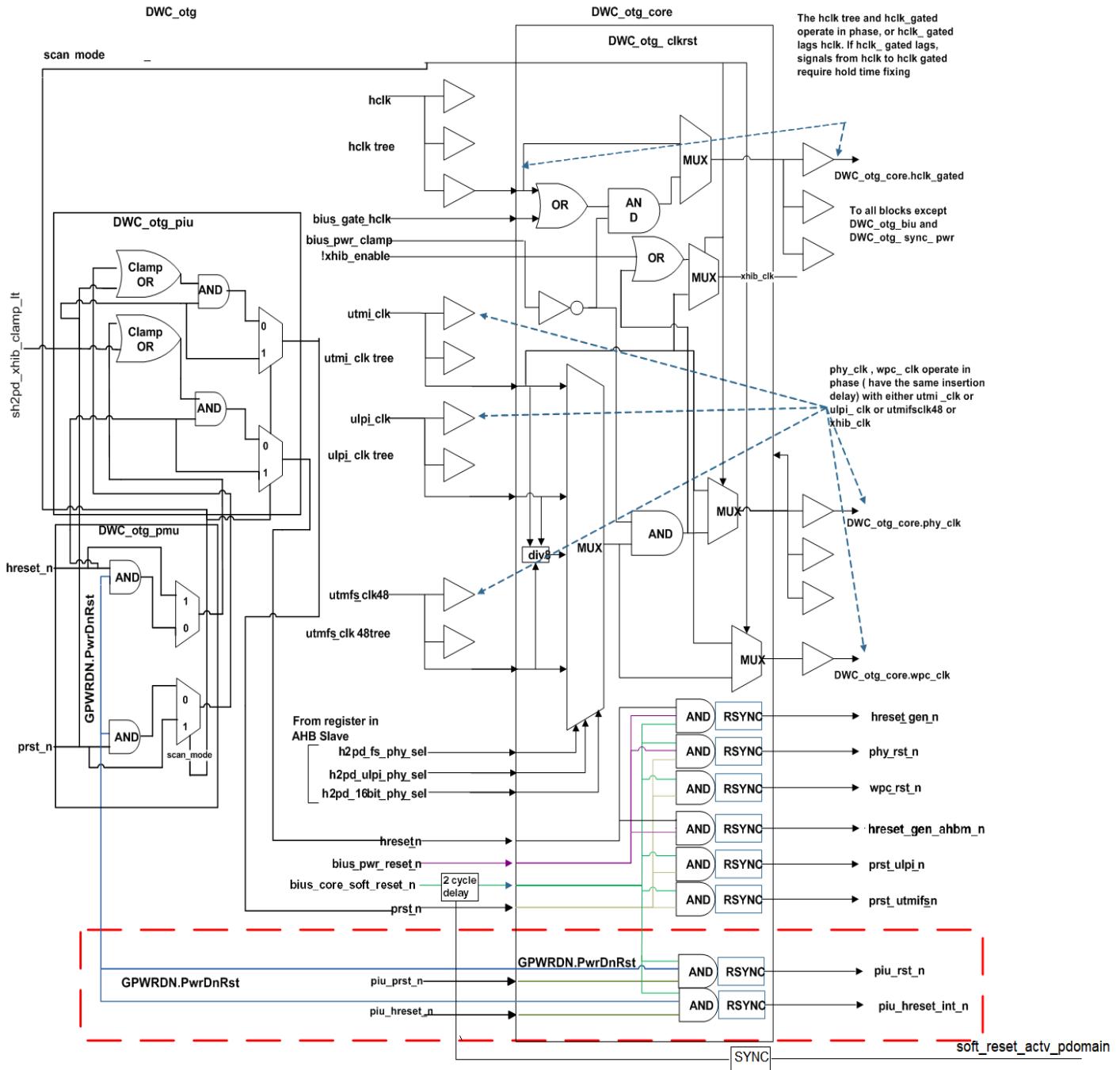


Figure 2-34 shows the logic for system clocks and reset generation when extended hibernation is enabled.

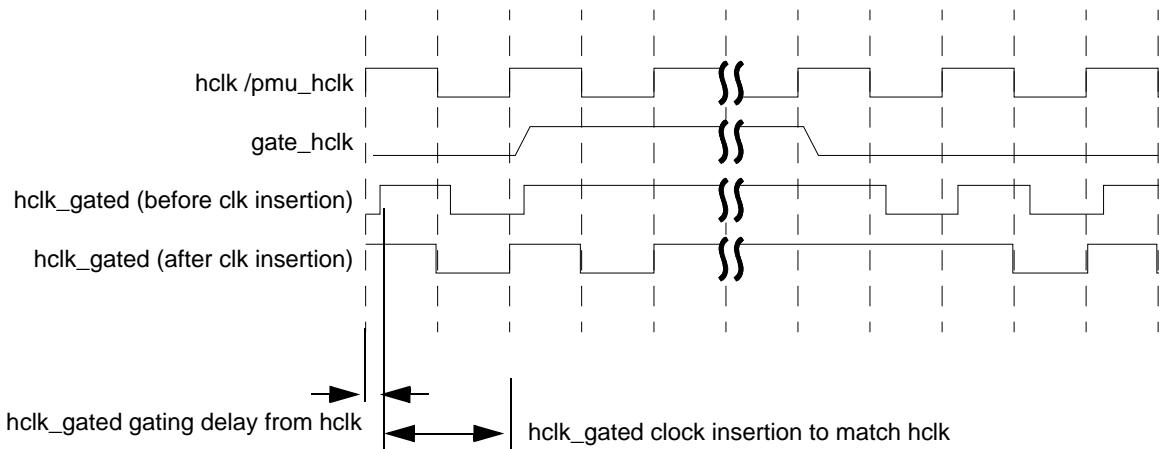
**Figure 2-34 System Clocks and Reset When Extended Hibernation is Enabled**



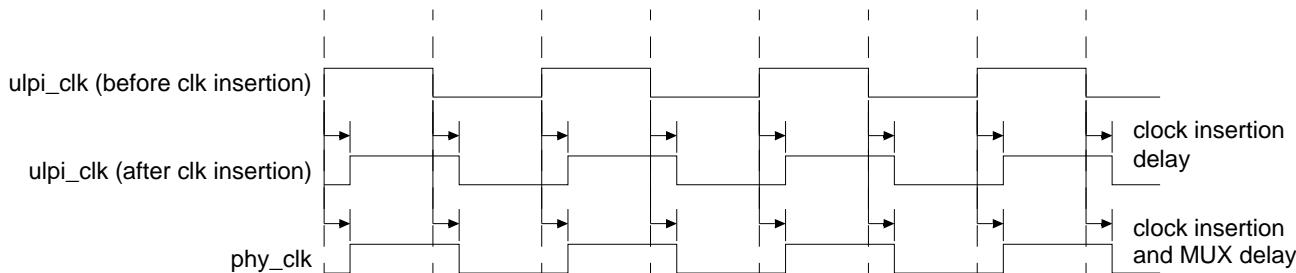
### 2.4.3 Clock Gating and Clock Tree Insertion

Figures 2-35 and 2-36 show AHB and PHY clock gating and clock tree insertion.

**Figure 2-35 AHB Clock Gating and Clock Tree Insertion**



**Figure 2-36 PHY Clock Tree Insertion**



Typically, the utmi\_clk or ulpi\_clk are PHY outputs, and all PHY outputs and inputs are timed to these clocks. Because phy\_clk is generated from either utmi\_clk or ulpi\_clk and goes to many flops in the DWC\_otg controller, it has a clock-tree insertion delay. All PHY outputs sampled by phy\_clk require input hold time fixes. PHY inputs driven by phy\_clk do not have hold time issues.

Because the gated clock signal, hclk\_gated, is the reference clock for multiple DWC\_otg controller flops, it has a clock tree insertion delay. If the hclk frequency in the system is fixed, you can delay the hclk\_gated clock tree to match hclk. This avoids hold time fixes between the hclk and hclk\_gated domains. hclk and hclk\_gated can also be matched through a PLL, when hclk is variable. hclk\_gated can also be delayed from hclk, using the hclk\_gated clock tree delay. This requires all signals going from the hclk domain to the hclk\_gated domain to have hold time fixes. This reduces the maximum operating AHB frequency, because the signals from hclk\_gated to hclk have less setup time.

## 2.4.4 System Resets

The OTG controller has two active low reset inputs:

- **hreset\_n**: hclk and pmu\_hclk domain resets
- **prst\_n**: phy\_clk domain reset

The following resets are generated in DWC\_otg\_pmu module if hibernation is enabled:

- **prst\_pwroff\_n**: prst\_n reset is ANDed with the hibernation reset in PMU module.
- **hreset\_pwroff\_n**: hreset\_n reset is ANDed with the hibernation reset in PMU module.

The following additional resets are inputs to the DWC\_otg controller if extended Hibernation is enabled:

- **piu\_prst\_n**: phy\_clock domain reset
- **piu\_hreset\_n**: hclk domain reset

In addition, the following resets are generated internally in the DWC\_otg\_clkrst module:

- **prst\_ulpi\_n**: prst\_n is synchronized to ulpi\_clk and ANDed with an internal software reset. Present only when ULPI PHY is selected. If hibernation is enabled, this reset is also ANDed with the hibernation reset.
- **prst\_utmifs\_n**: prst\_n is synchronized to utmifs\_clk48 and ANDed with an internal software reset. Present only when dedicated USB1.1 FS PHY interface is selected. If hibernation is enabled, this reset is also ANDed with the hibernation reset.
- **wpc\_rst\_n**: prst\_n is synchronized to phy\_clk and ANDed with an internal software reset. Present only when Power Optimization is enabled. If hibernation is enabled, this reset is also ANDed with the hibernation reset.
- **phy\_rst\_n**: prst\_n is synchronized to phy\_clk and ANDed with internal software and power-down resets. If hibernation is enabled, this reset is also ANDed with the hibernation reset.
- **hreset\_gen\_n**: hreset\_n is ANDed with internal software and power-down resets. If hibernation is enabled, this reset is also ANDed with the hibernation reset.
- **hreset\_gen\_ahbm\_n**: hreset\_n is ANDed with power-down resets. If hibernation is enabled, this reset is also ANDed with the hibernation reset.



**Note** All input resets to the DWC\_otg controller can be asserted asynchronously, but deassertion must be synchronous with respect to the clock selected in the configuration.

Whenever the software changes the PHY selection bits (USB 1.1 FS transceiver, UTMI+, or ULPI) after reset, it must assert a software reset, because the PHY section bits change the clock to the PHY Interface Unit.

You can choose between Synchronous and Asynchronous reset modes during coreConsultant configuration by specifying the Reset Style of Clocked always Blocks in RTL. In RTL, all “clocked-always” blocks are coded for both asynchronous reset and synchronous reset, as shown in the following code. The reset is an active-low signal.

The DWC\_otg\_HCLK\_RST\_MODE is defined as “or negedge hreset\_n” or as “” depending on whether you select asynchronous or synchronous resets in coreConsultant.

```

always @ (posedge hclk 'DWC_otg_HCLK_RST_MODE) begin
begin
  if(~hreset_n)
    begin
      .....
    end
  else
    begin
      .....
    end
end

```

Even when you select synchronous reset, the following flops are asynchronously reset. These flops control the PHY interface and must be in a known state for the PHY to start the clock.

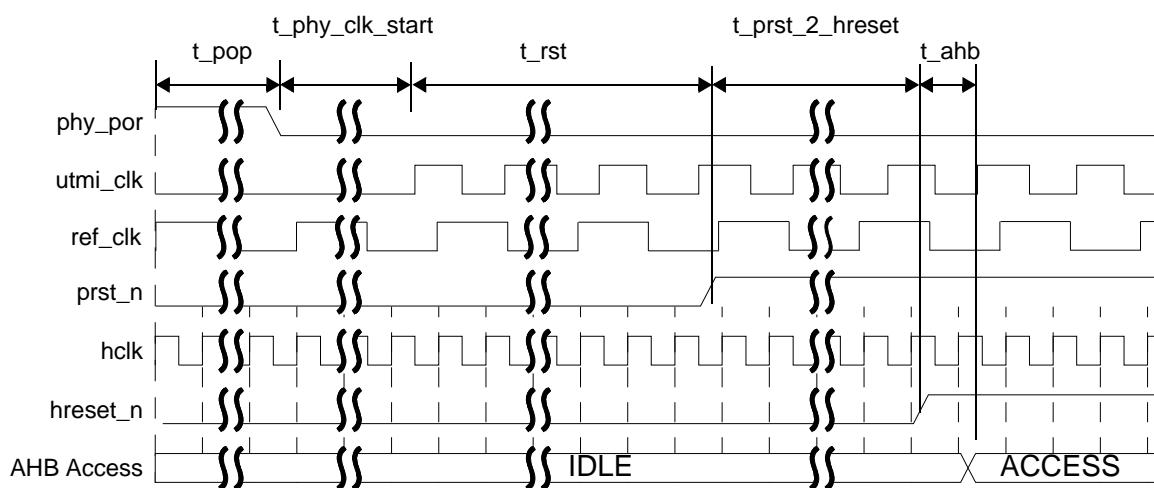
- DWC\_otg/U\_DWC\_otg\_wpc/wpc\_suspend\_n

In addition, the following signals are MUXed to a valid state during reset, and thus have a combinational path with respect to reset:

- DWC\_otg/utmi\_xcvrselect
- DWC\_otg/utmi\_opmode
- DWC\_otg/utmi\_termselect

To meet timing requirements and operate correctly, the hreset\_n signal must be deasserted synchronous to the hclk signal and can be asynchronously asserted. Asynchronous reset assertion guarantees that all signals and drivers are in a safe state during power-up when the clocks are typically not active or are unstable. Synchronous reset deassertion guarantees that there are no timing or metastability violations.

**Figure 2-37 Clock and Reset Requirements**



For Extended Hibernation configurations (OTG\_EN\_PWROPT = 3), the behavior of piu\_hclk is similar to the hclk behavior. Similarly, the behavior of piu\_prst\_n is similar to the prst\_n behavior.

Note the following details for [Figure 2-37](#):

- **t\_por:** PHY power-on reset time
- **t\_phy\_clk\_start:** PHY reset removal to clock start (typically in microseconds)
- **t\_RST:** DWC\_otg PHY clock domain reset and AHB hclk domain reset over lap time (a minimum of 12 cycles of the slowest clock is recommended.)
- **t\_prst\_2\_hreset:** prst\_n removal to hreset\_n remove (a minimum of 6 cycles of the slowest clock is recommended.)
- **t\_ahb:** hreset\_n removal to AHB access start (1 clock)
- **prst\_n and hreset\_n:** The delay between prst\_n and hrst\_n must not be larger than iddig filter time.

Overlap is required between PHY and AHB domain reset time (**t\_RST**) so that interface signals between the two domains are correctly reset.

Setting the **t\_prst\_2\_hreset** delay or setting **t\_ahb** to 6 clocks is recommended to ensure that the PHY domain is ready immediately after reset.

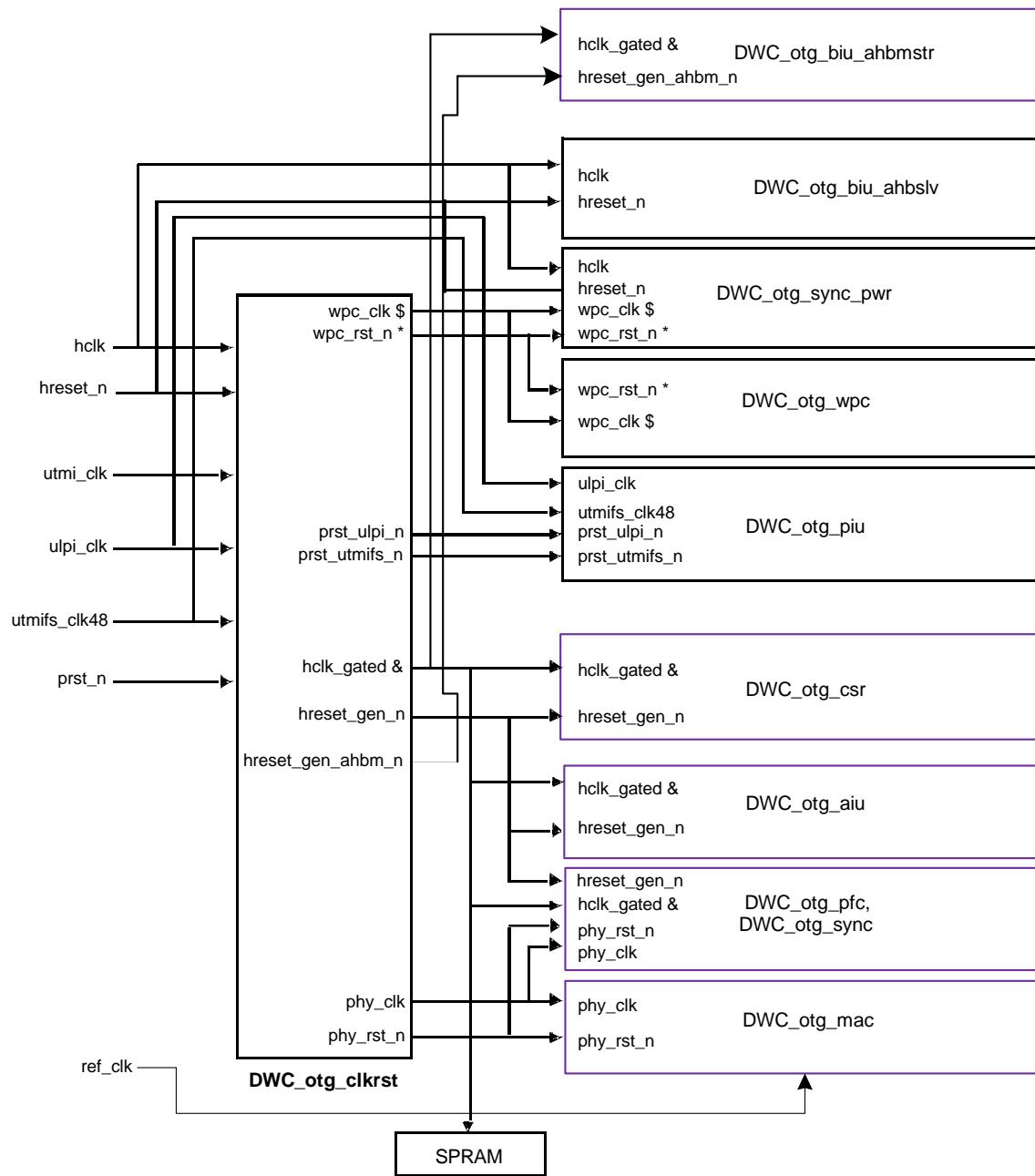


Because DWC\_otg supports software-selectable PHY options, after software select a PHY, it must issue a soft reset.

## 2.4.5 Clock and Reset Distribution Without PMU

Figure 2-38 shows the controller's clock and reset distribution logic when the PMU is not present.

**Figure 2-38 Clock and Reset Distribution Without PMU**



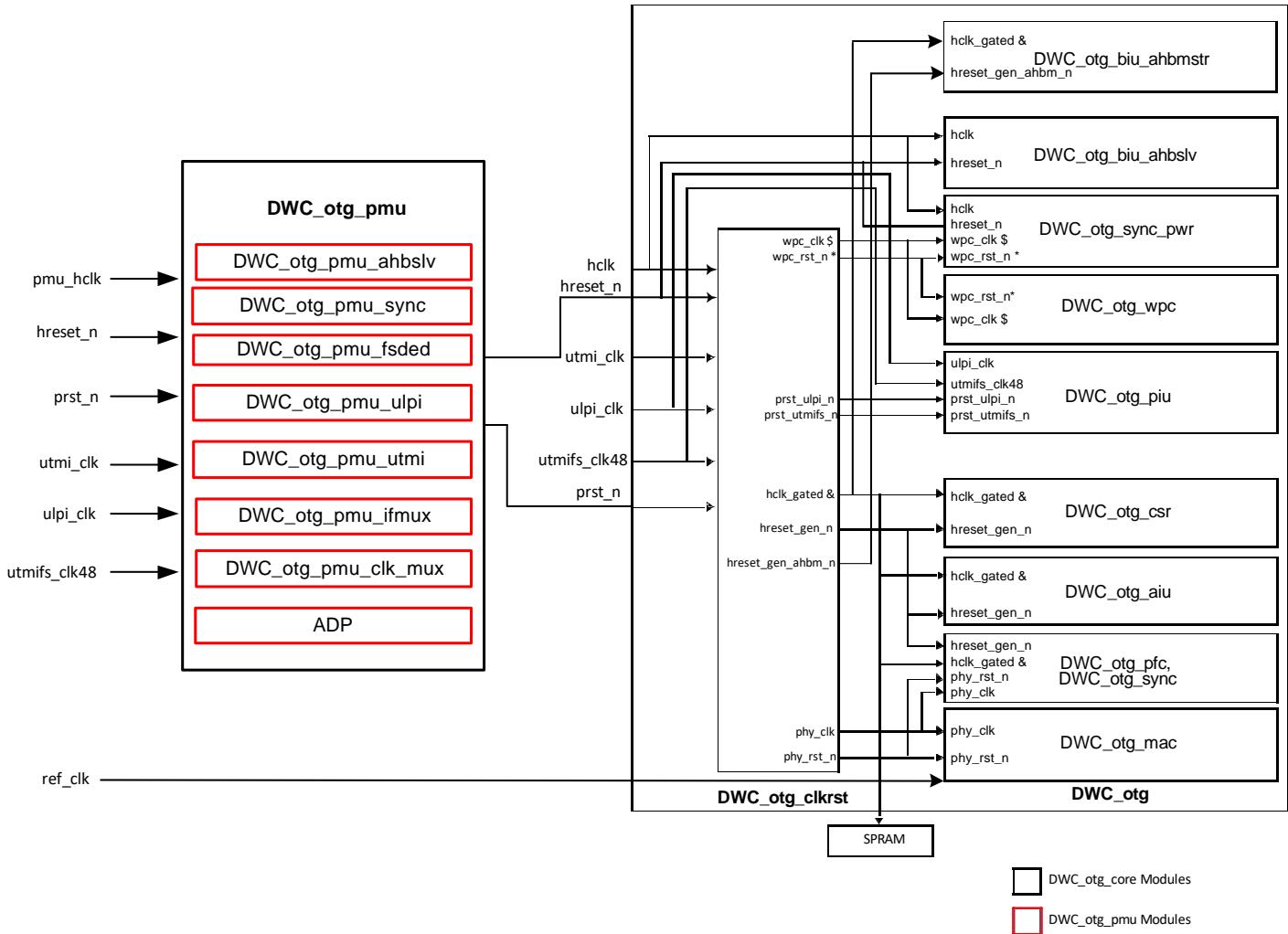
When OTG\_EN\_PWR0PT==0,

- \* - phy\_rst\_n is used instead of wpc\_rst\_n
- \$ - phy\_clk is used instead of wpc\_clk
- & - hclk is used instead of hclk\_gated

## 2.4.6 Clock and Reset Distribution With PMU

Figure 2-39 shows the clocks and reset distribution when PMU is present.

**Figure 2-39 Clock and Reset Distribution With PMU**



When hibernation is enabled, the clock multiplexor ensures that all the phy domain logic is independent of the interface type by having reuse, thus reducing the gate count. The clock priority depends upon the selected signal. For example:

1. If SCAN mode is selected, scan\_clock has higher priority.
2. If SCAN mode is not selected, but FS and ULPI are selected, then utmifs\_clk48 has higher priority over ulpi\_clk. If only FS is selected, utmifs\_clk48 will be taken. Similarly, if only ulpi is selected, ulpi\_clk will be taken.
3. If none of the above are selected, utmi\_clk will be considered.

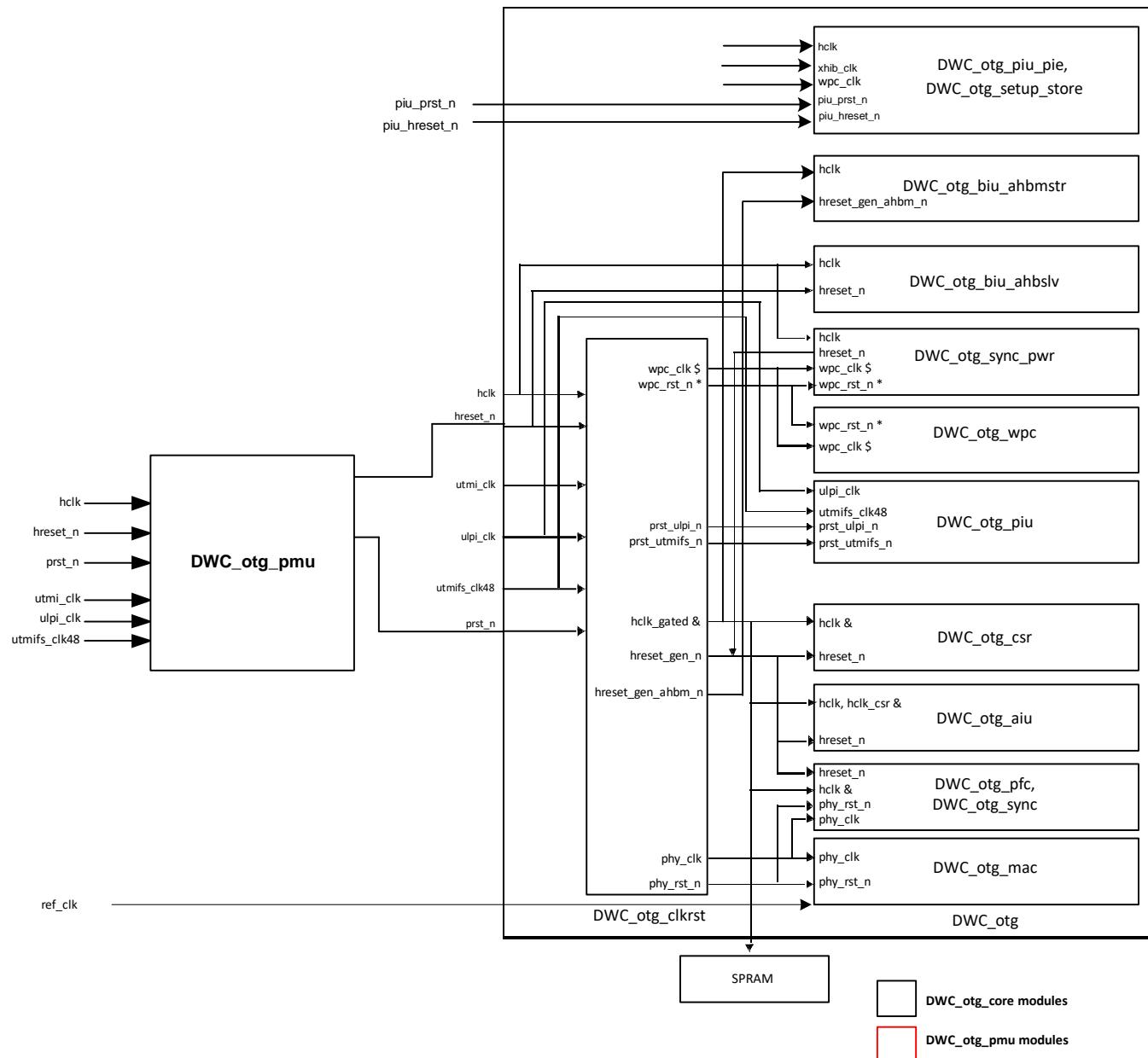
During final clock tree synthesis, or whenever the clock tree is being balanced, you must ensure the following clocks are balanced in order to meet the timing:

- The phy\_clk in DWC\_otg\_pmu must be balanced with the wpc\_clk and phy\_clk clocks in DWC\_otg\_core.
- Similarly, hclk in DWC\_otg\_pmu must be balanced with the hclk and hclk\_gated clocks in DWC\_otg\_core.

#### 2.4.7 Clock and Reset Distribution With Extended Hibernation

Figure 2-40 shows the controller's clock and reset distribution logic when extended hibernation is enabled.

**Figure 2-40 Clock and Reset Distribution When Extended Hibernation is Enabled**



## 2.4.8 Clock Domain Crossing

Two top-level modules handle clock synchronization of signals that cross clock domains:

- DWC\_otg\_sync.v: All signals that cross from the AHB clock (hclk/hclk\_gated) domain to the PHY clock (phy\_clk) domain, and vice-versa, pass through this module. When power optimization is enabled (OTG\_EN\_PWR0PT = 1), this module is off during Power Down mode.
- DWC\_otg\_sync\_pwr.v: All signals that cross from the AHB clock (hclk) domain to the AHB clock (wpc\_clk) domain, and vice-versa, pass through this module. When power optimization is enabled, this module is on during power-down mode.

In addition, inside the DWC\_otg\_pfc module, the DWC\_otg\_pfc\_sync.v module handles clock domain crossing within the PFC module. The PFC block itself is a two-clock-domain module.

The two-clock FIFO controller, DWC\_otg\_2clkfifo.v, has multi-bit, double synchronizers for passing the gray-coded FIFO pointers from one clock domain to another.

The inputs and outputs to these clock domain-crossing modules are prefixed as follows:

- h2pl\_: hclk domain to phy\_clk domain signals
- sh2pl\_: h2pl\_ signals synchronized to phy\_clk domain after double-flopping
- h2pt\_: hclk domain to phy\_clk domain toggle signals
- sh2pt\_: h2pt\_ signals synchronized to phy\_clk domain after triple-flopping and the last two stages XORed to generate a 1-clock active high signal
- h2pd\_: hclk domain to phy\_clk domain signals that require no synchronization. These signals do not require synchronization, because they are data signals that are stable when sampled, and qualified with corresponding control signals.
- sh2pd\_: Same as h2pd\_ signals, but renamed for identification. These non-synchronized signals are passed through the synchronizer for design clarity to identify clock domain-crossing signals.
- h2pb\_: hclk domain to phy\_clk domain multi-bit bus signals. Binary-to-gray conversion and multi-bit double synchronizers cannot be used to synchronize these signals, because bit-at-a-time changes are not guaranteed.
- sh2pb\_: h2pl\_ signals synchronized to phy\_clk domain using a bus synchronizer. Whenever any bit on the bus changes, the bus is synchronized to the phy\_clk domain.

Signals that cross from phy\_clk to hclk are prefixed as: p2hl\_, sp2hl\_, p2ht\_, sp2ht\_, p2hd\_, sp2hd\_, p2hb\_, and sp2hb\_.

Signals that cross the ref\_clk domain are prefixed as: r2hl\_, sr2hl\_, h2rd\_, sh2rd\_, p2rl\_, sp2rl\_, r2pl\_, sr2pl\_, and p2rt\_sp2rt\_.

The clock frequencies used for testing functional simulations is as follows:

- AHB clock frequencies (AHB clock (HCLK)):
  - 50 MHz
  - 52 MHz
  - 60 MHz

- 100 MHz
- 150 MHz
- 166 MHz



**Note** VTB is tested for 60 MHz. Other frequencies are tested in CRV.

- PHY clock frequencies (PHY CLK)
  - 30 MHz
  - 60 MHz
  - 48 MHz
- REF clock frequencies (ref\_clk)
  - 12 MHz
  - 16 MHz
  - 17 MHz
  - 19.2 MHz
  - 20 MHz
  - 24 MHz
  - 30 MHz
  - 40 MHz

### Identifying Metastability Paths in Gate-Level Simulation

A simple DWC\_otg\_bcm21 which can be configured as a double synchronizer or a multi-bit, double synchronizer is instantiated inside the synchronizers, clock generator, and 2-clock FIFOs. When you perform gate-level simulation, disable setup and hold timing checks on the first DFF of the synchronizers.

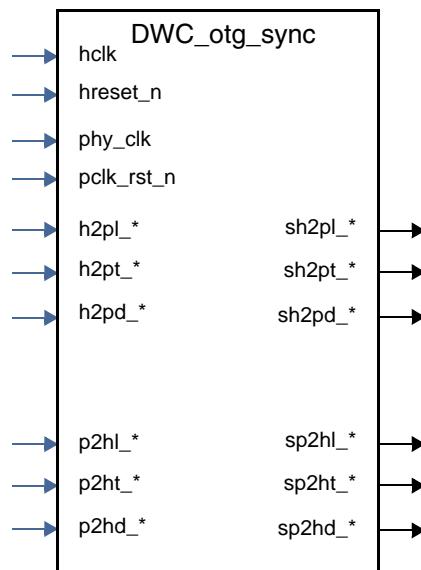
In DWC\_otg\_bcm21, the name of the first DFF in the synchronizer is sample\_meta. To get a list of all these DFFs, grep DFF by name after synthesis, as shown in the following example:

1. Perform synthesis with the target library (as usual) using Design Compiler.
2. Run the following command:

```
foreach_in_collection itr [get_cell -h *sample_meta*reg*] { echo "[get_attribute $itr full_name]"; } > sync_cell.rpt
```

3. Review sync\_cell.rpt.

Figure 2-41 on page 125 shows the inputs and outputs of the DWC\_otg\_sync block.

**Figure 2-41 DWC\_otg\_sync Clock Synchronization Block**

[Figure 2-42](#) shows the level and toggle synchronization logic.

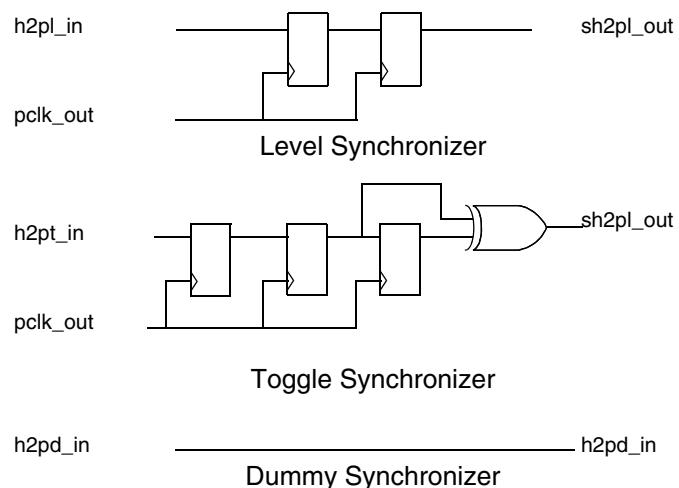
**Figure 2-42 Level, Toggle, and Dummy Synchronizer**

Figure 2-43 shows the bus synchronization logic.

**Figure 2-43 Bus Synchronizer Block Diagram**

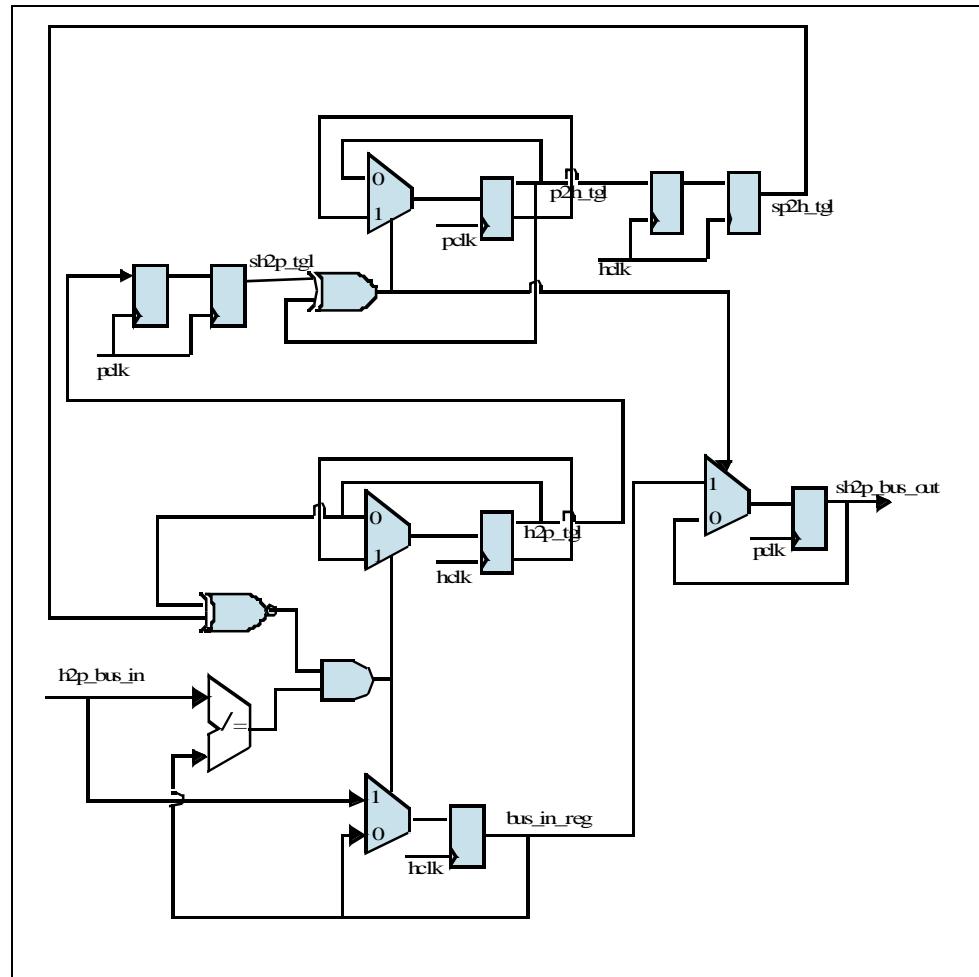
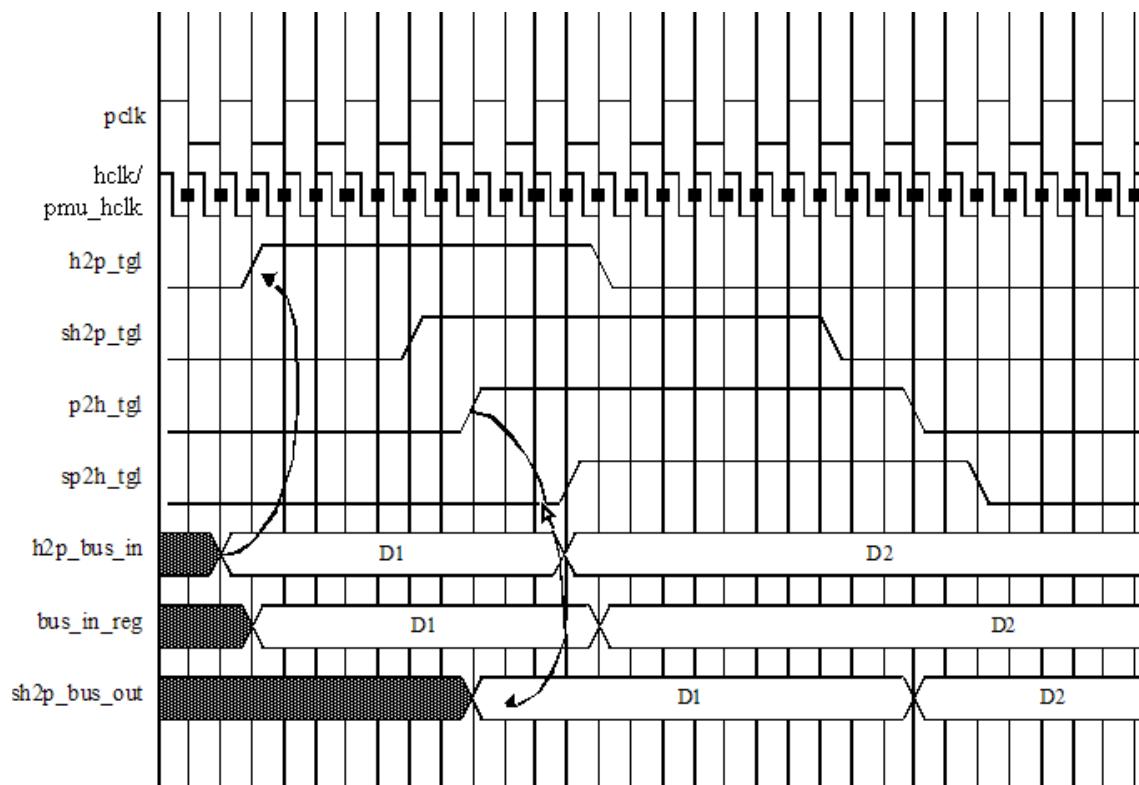


Figure 2-44 shows the timing diagram for h2p bus synchronizer.

h2p\_tgl is the toggle signal generated when there is a change in the data bus input. This is synchronized in phy\_clk domain (sh2p\_tgl). p2h\_tgl is generated in phy\_clk domain when it has taken the h2p\_bus\_in change. This is synchronized back to hclk domain s(p2h\_tgl).

**Figure 2-44 Bus Synchronizer Timing Diagram**



## 2.5 Interface and Protocol Timing

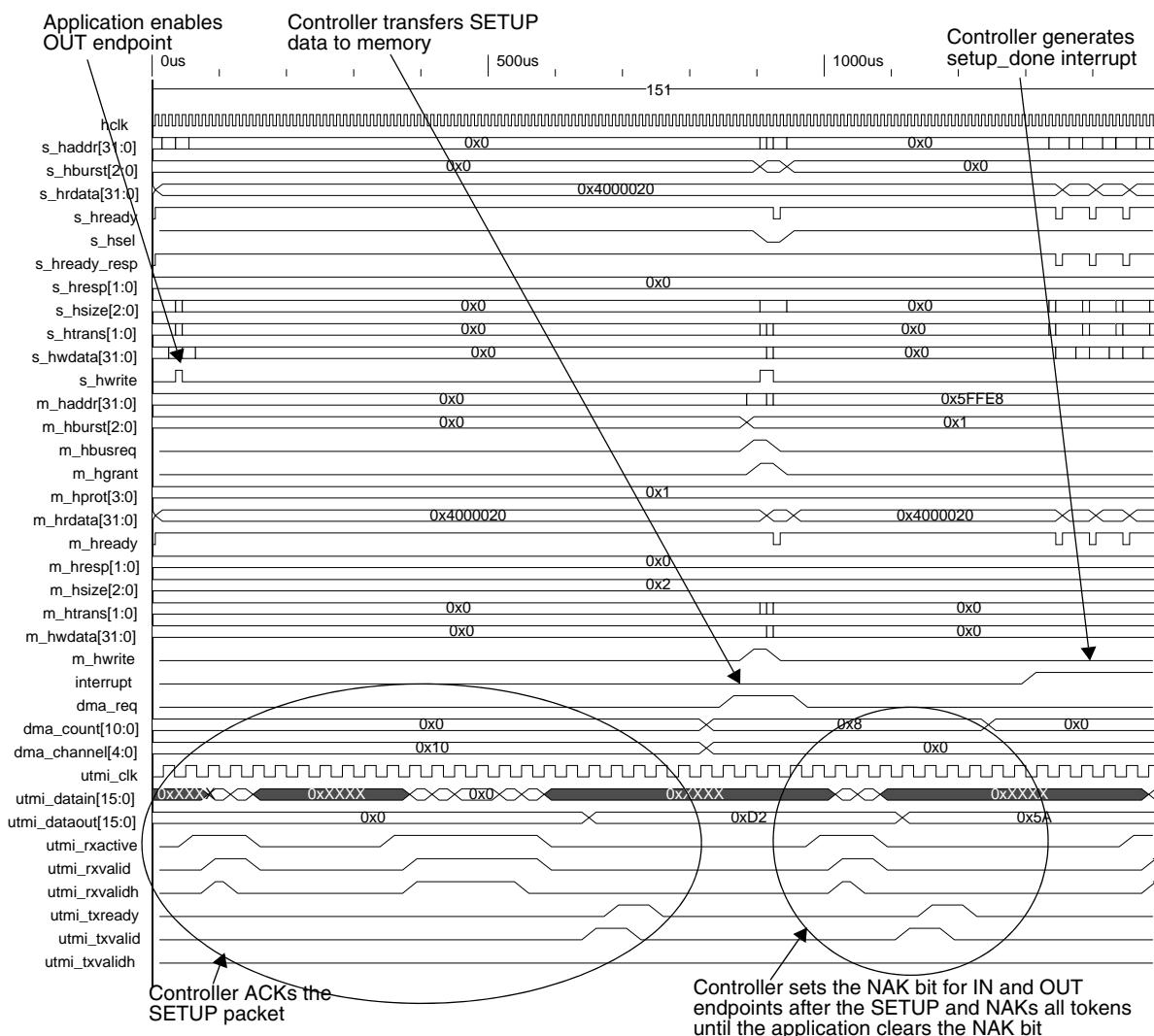
This section describes the following topics:

- “Control Write” on page 128
- “Device Mode Bulk OUT” on page 131
- “Device Mode Bulk IN” on page 132
- “Device Mode Interrupt OUT” on page 133
- “Device Mode Isochronous IN” on page 134
- “Host Mode Isochronous IN” on page 136
- “Host Mode Bulk Out in Slave Mode” on page 138

### 2.5.1 Control Write

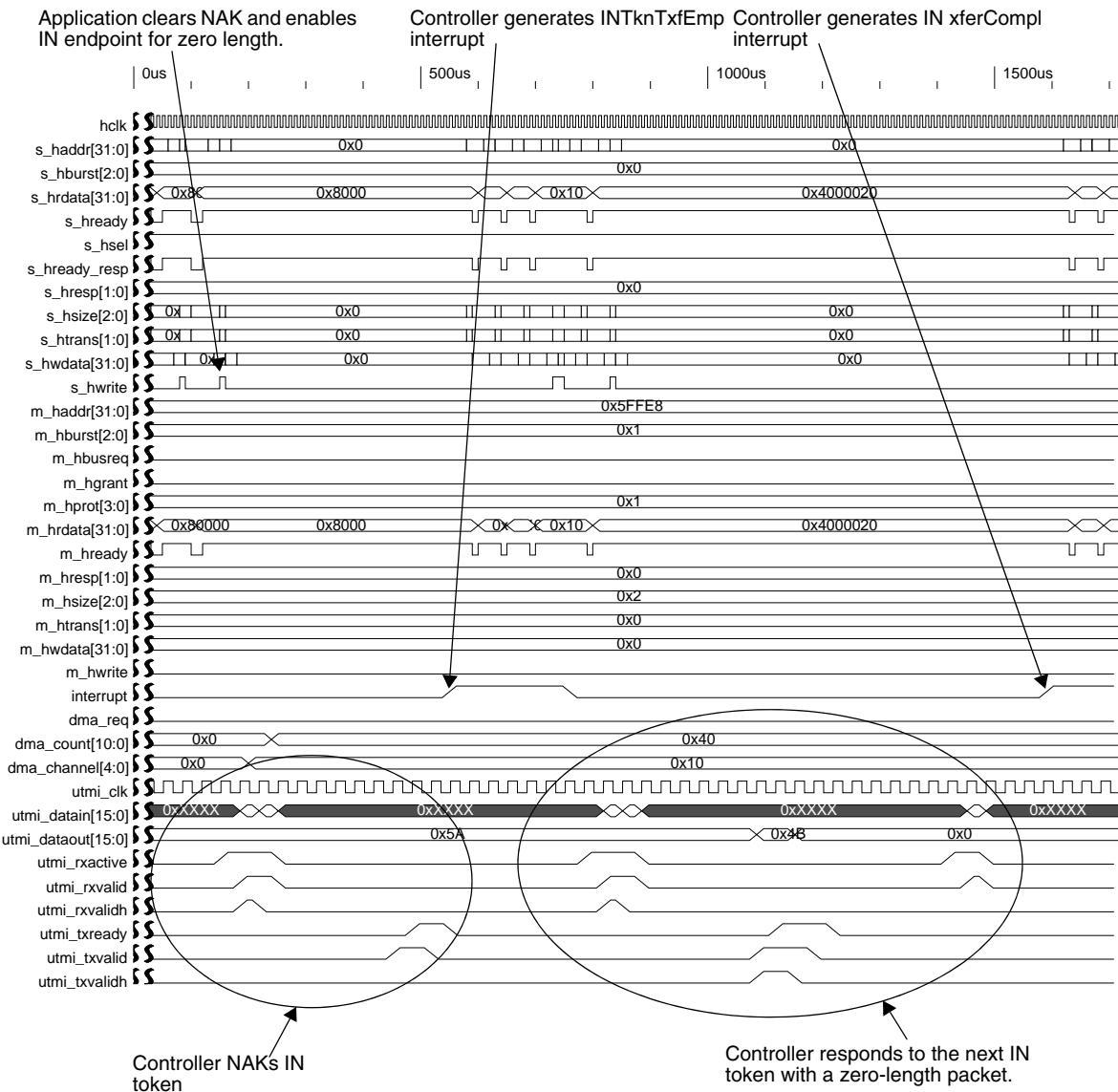
Figures 2-45 and 2-46 show how the controller handles control writes, such as the SetAddress request (command), in Device mode. This example shows the controller in DMA mode with a 16-bit UTMI interface. There is no Data stage in this control transfer, which has a zero-length Status stage.

The controller ACKs the SETUP packet and sets NAK on both IN and OUT endpoints. All tokens to that endpoint receive a NAK response until the application sets up the endpoint and clears the NAK. Figure 2-45 shows the Setup stage.

**Figure 2-45 Setup Phase for SetAddress Request Using DMA Mode**

**Figure 2-46** shows the Status stage. The controller gives data only after the application sets up the IN endpoint and clears the NAK.

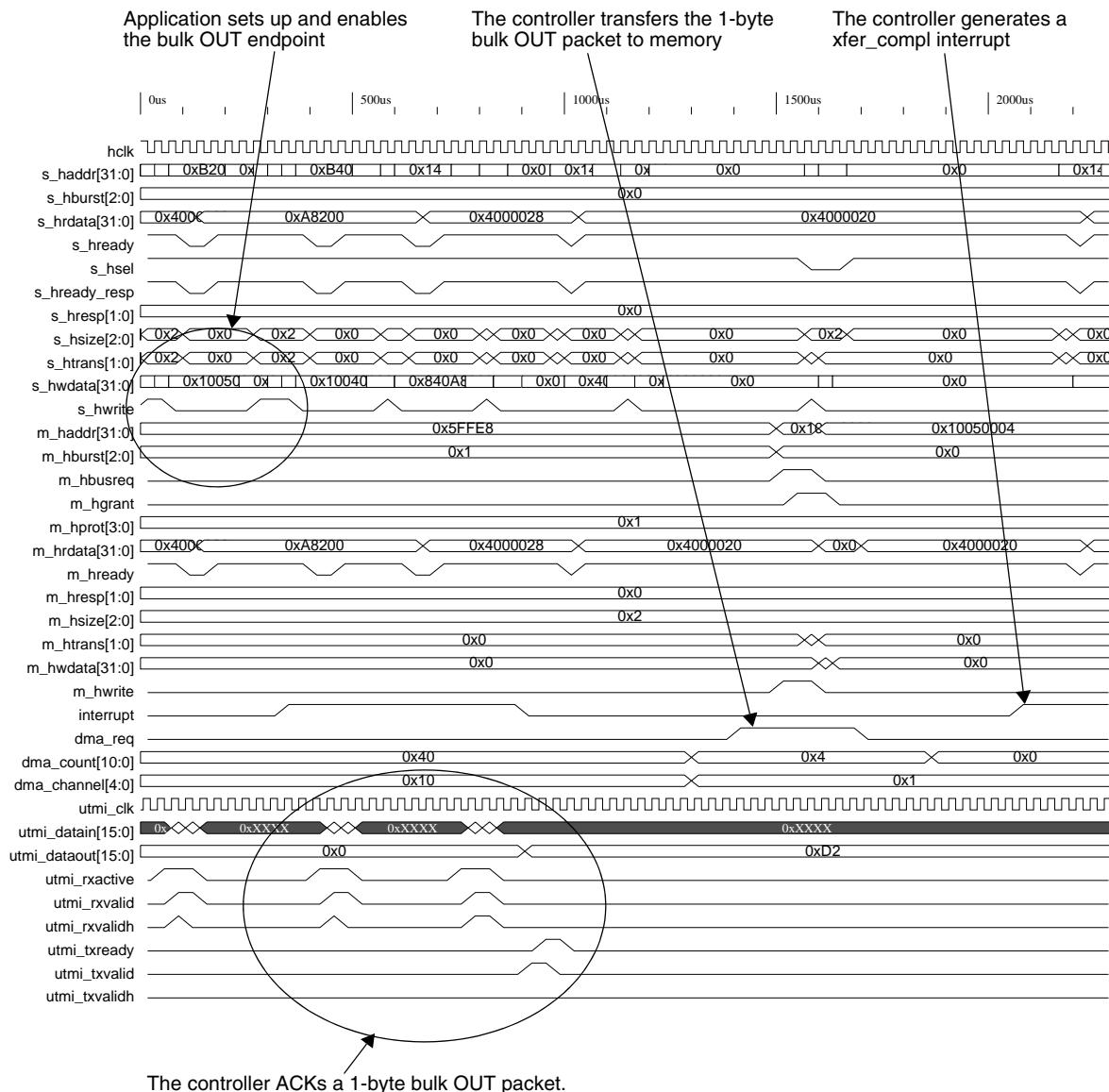
**Figure 2-46 Status Phase of SetAddress Request Using DMA Mode**



## 2.5.2 Device Mode Bulk OUT

Figure 2-47 shows a Device mode bulk OUT transaction in DMA mode for a 1-byte packet. The application sets up the endpoint and waits for the xfer\_compl interrupt.

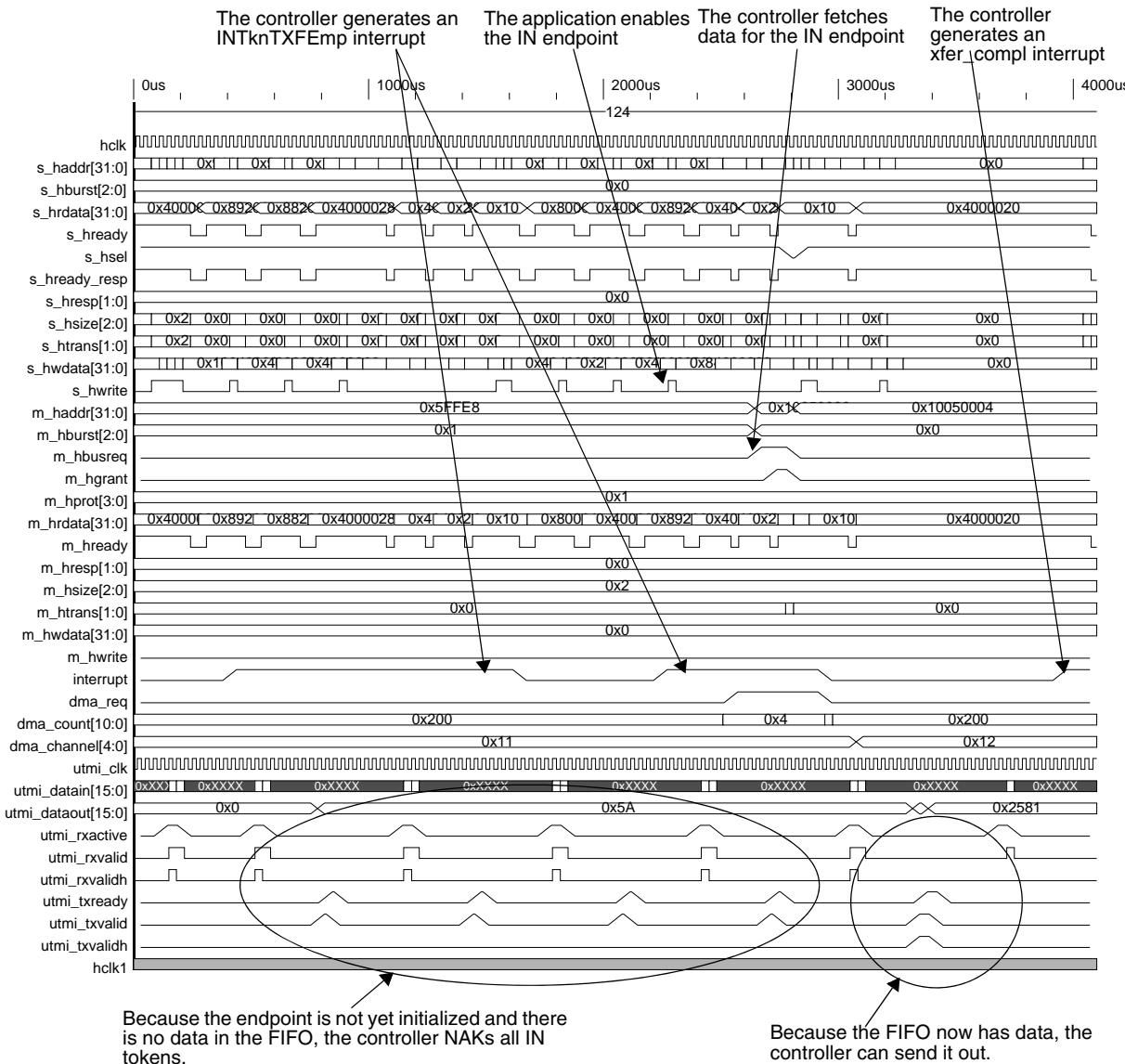
**Figure 2-47 Device Bulk OUT Transaction Using DMA Mode**



### 2.5.3 Device Mode Bulk IN

Figure 2-48 shows a Device mode bulk IN transaction in DMA mode. This example is for a 1-byte IN packet.

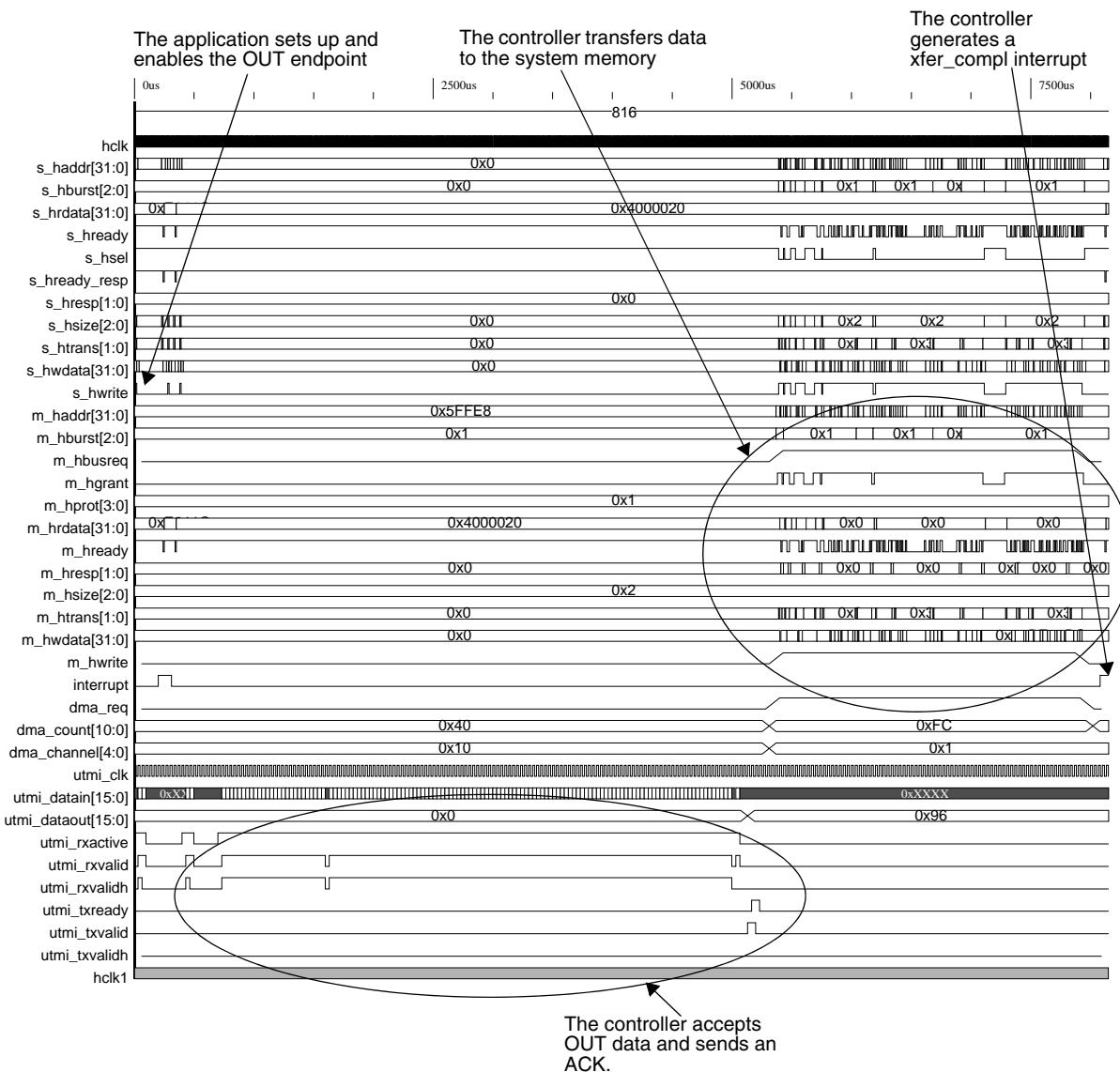
**Figure 2-48 Device Bulk IN Transaction Using DMA Mode**



#### **2.5.4 Device Mode Interrupt OUT**

[Figure 2-49](#) shows a Device mode interrupt OUT transaction in DMA mode. In Device mode, this is very similar to a bulk transaction. The example shown here is for a 252-byte packet.

**Figure 2-49 Device Interrupt OUT Transaction Using DMA Mode**

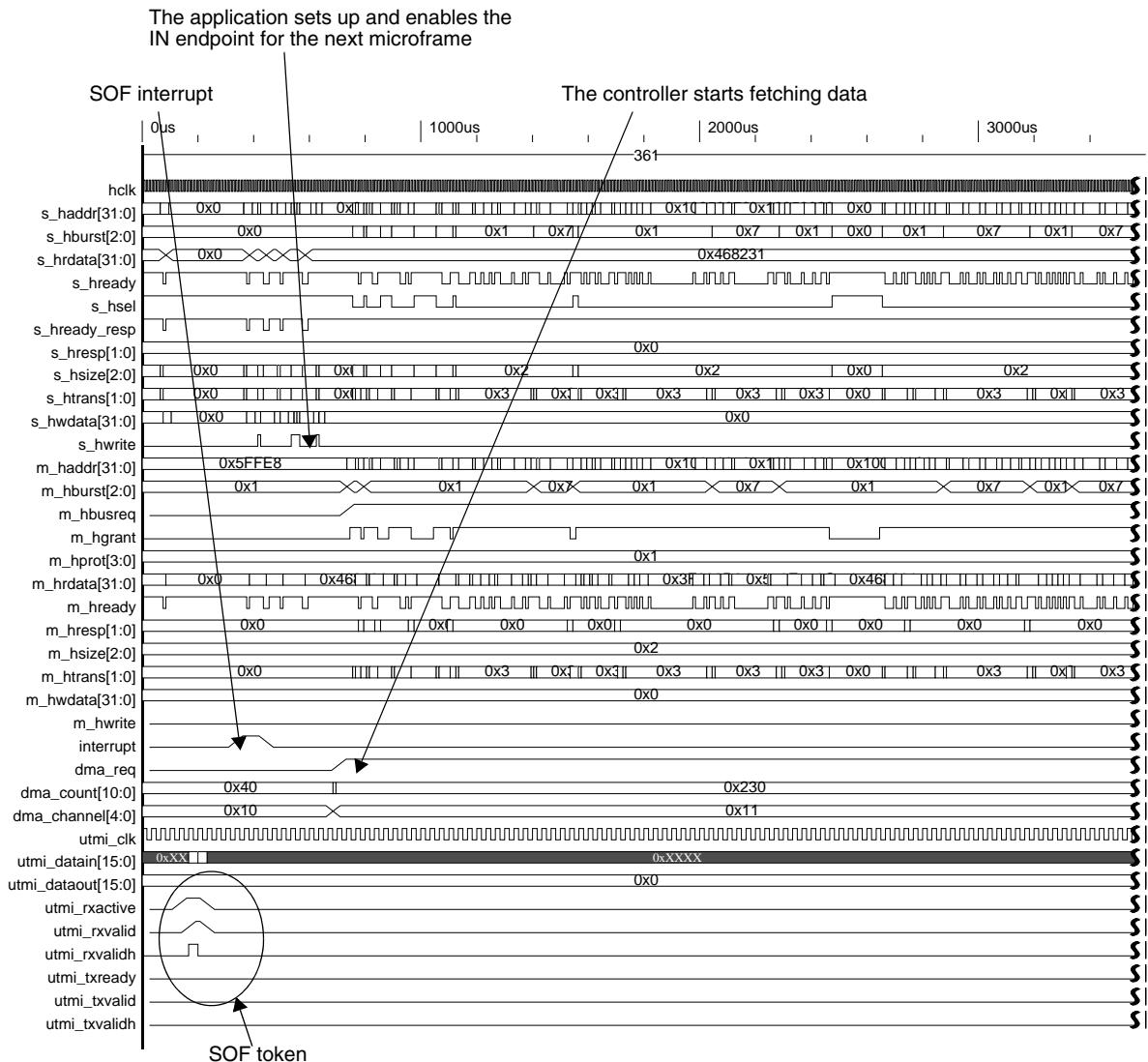


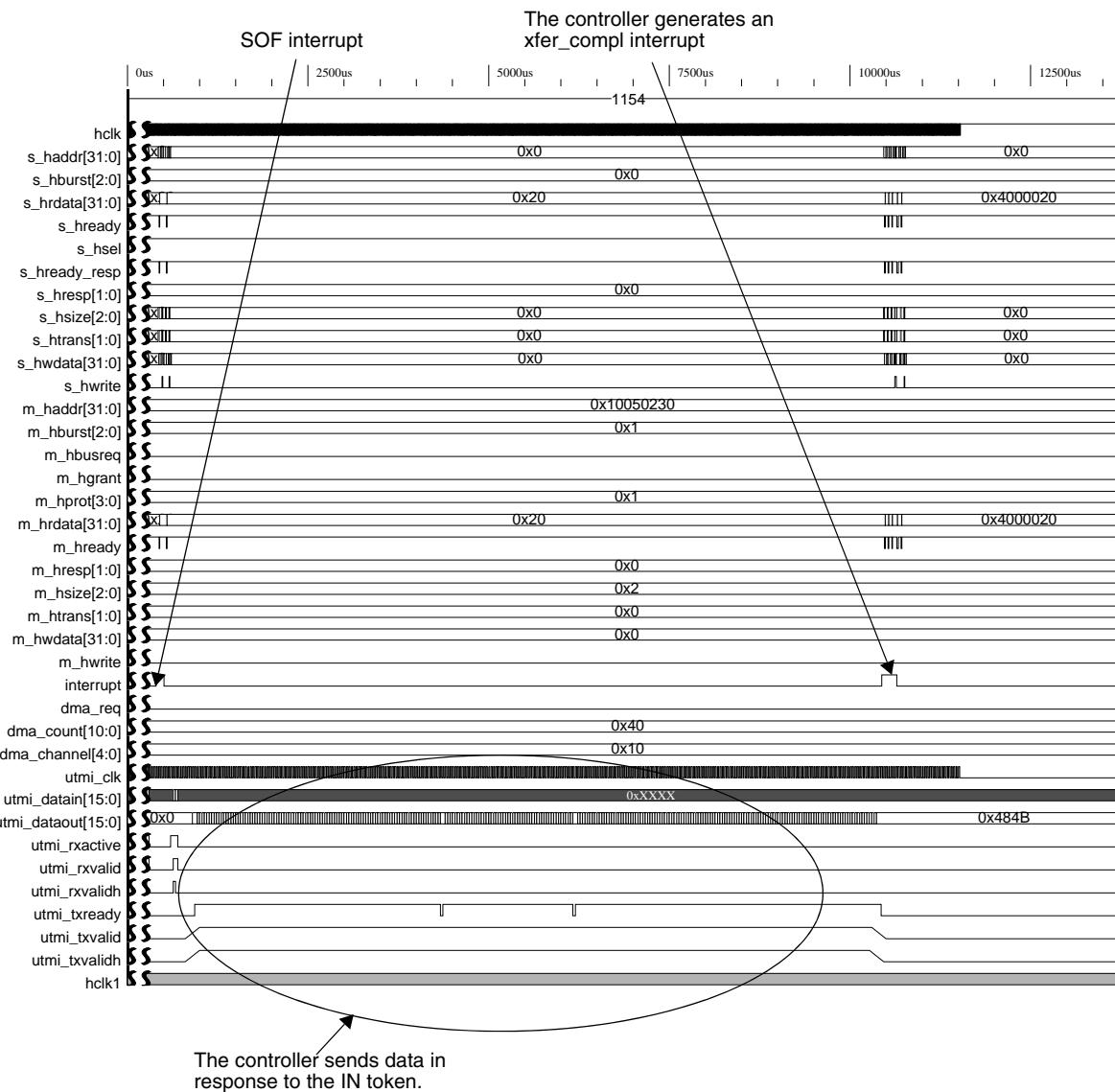
## 2.5.5 Device Mode Isochronous IN

Figure 2-50 shows a Device mode isochronous IN transaction in DMA mode. The application schedules the transfer for the next microframe and the controller fetches the data once the endpoint has been enabled.

Figure 2-51 on page 135 shows the controller sending this data in the next microframe, in response to an IN token for this endpoint.

**Figure 2-50 Device Isochronous IN Transaction Using DMA Mode (1 of 2)**



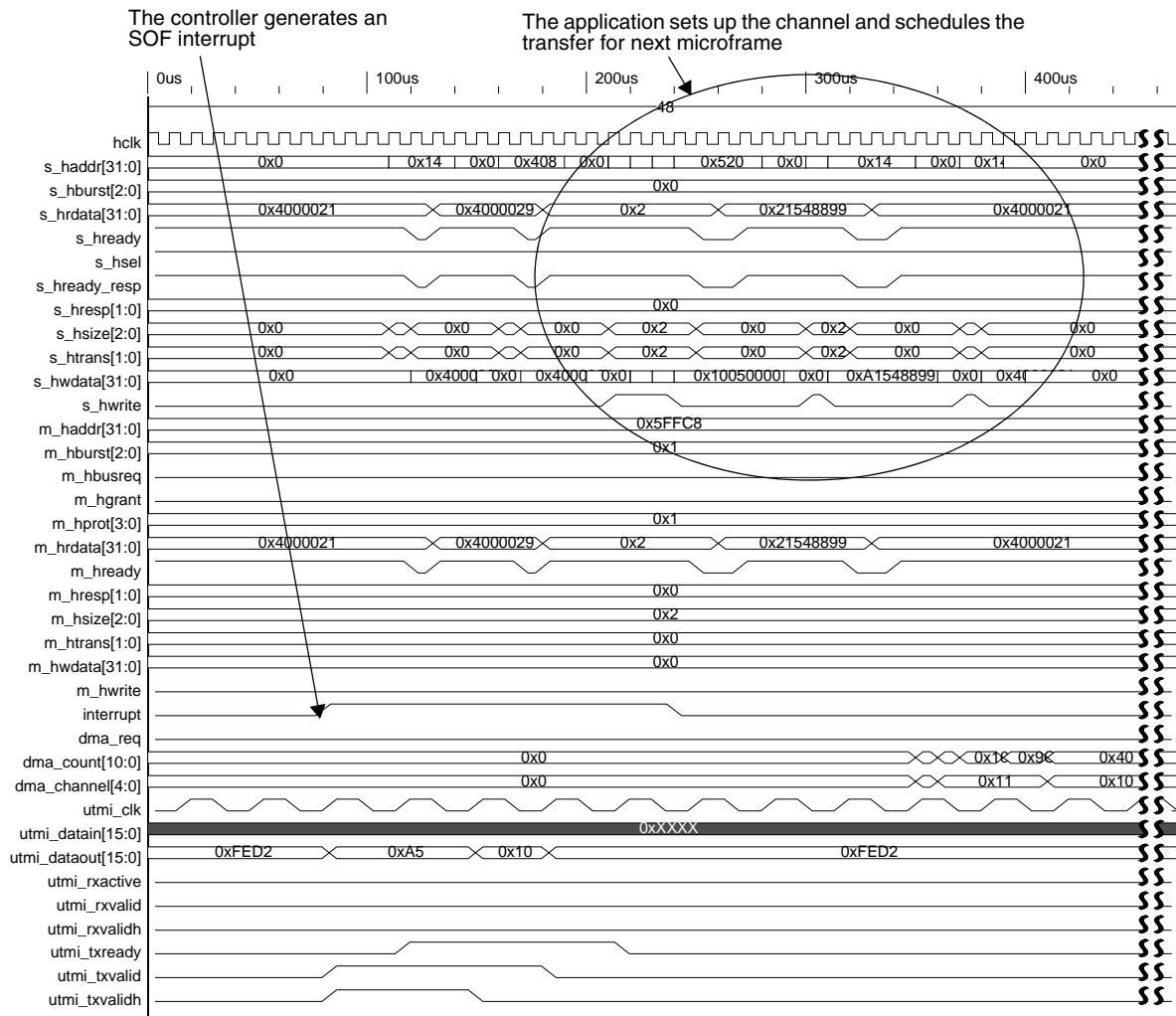
**Figure 2-51 Device Isochronous IN Transaction Using DMA Mode (2 of 2)**

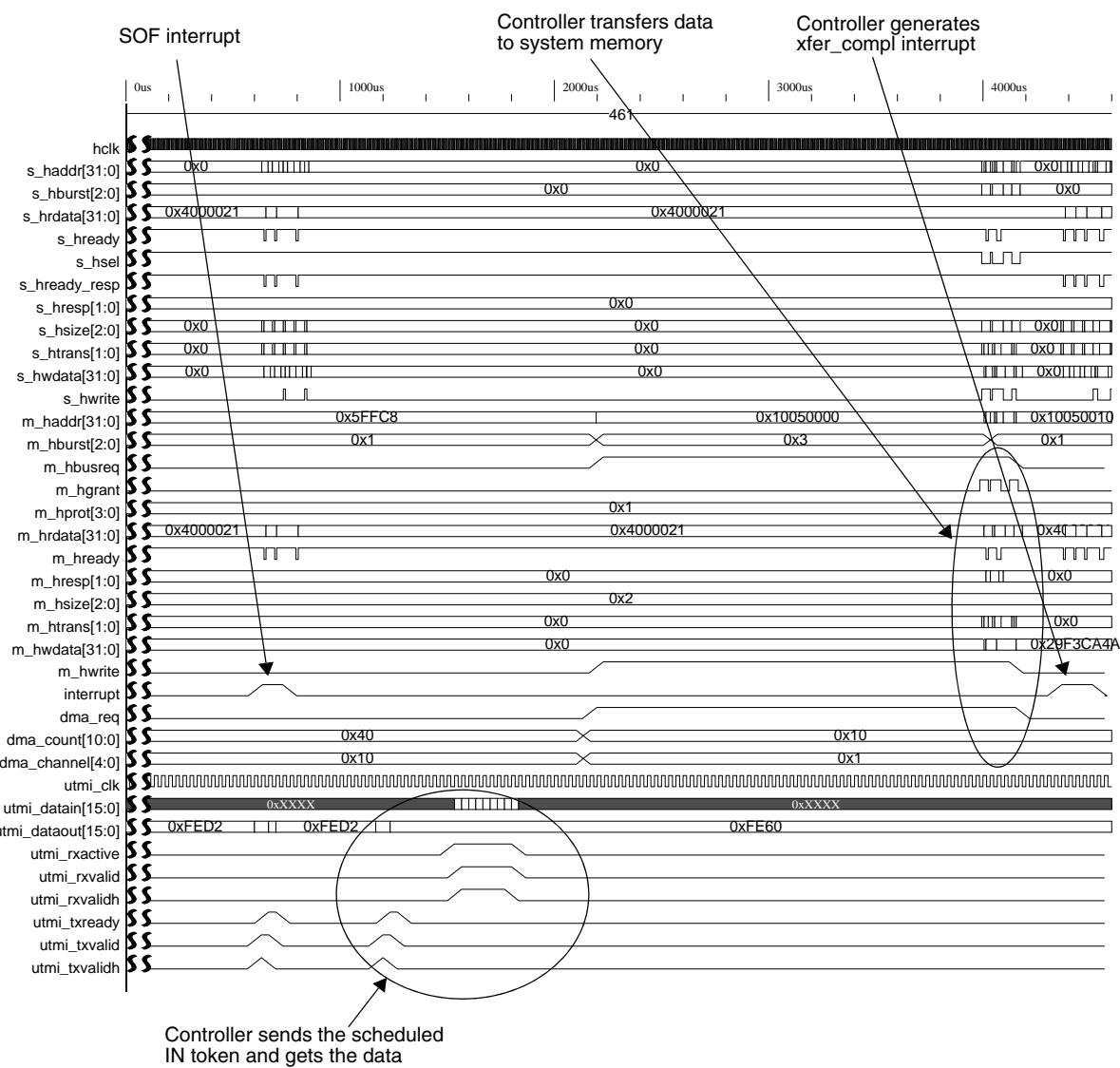
## 2.5.6 Host Mode Isochronous IN

Figures 2-52 and 2-53 show a Host mode isochronous IN transaction in DMA mode. The application must schedule the transfer one (micro) frame before the transfer.

Figure 2-52 shows how the application schedules isochronous IN transfers one microframe ahead of the transfer. Figure 2-53 on page 137 shows how the controller performs the transfer in the scheduled (next) microframe.

**Figure 2-52 Host Isochronous IN Scheduling Using DMA Mode**

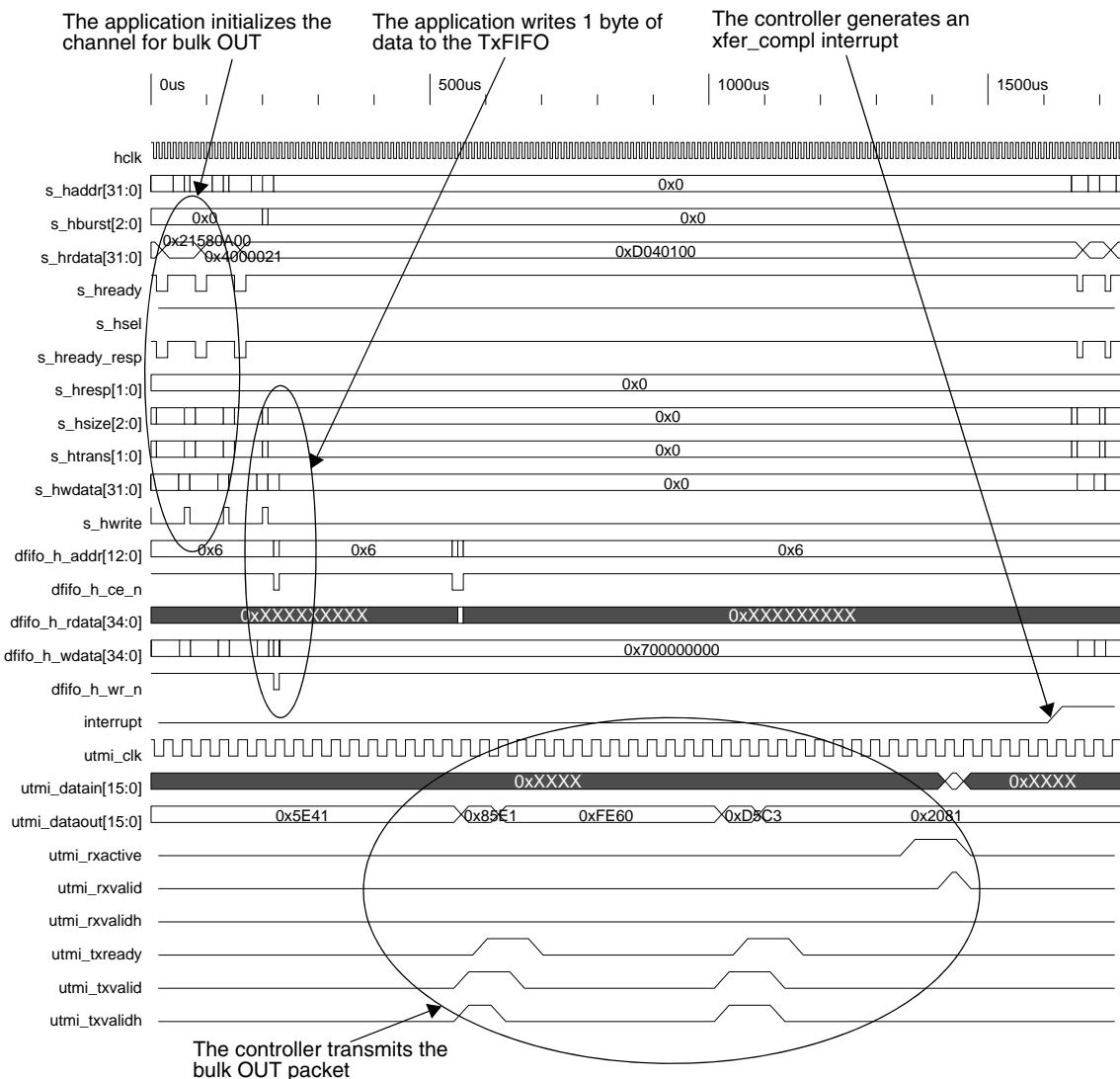


**Figure 2-53 Host Isochronous IN Transaction Using DMA Mode**

## 2.5.7 Host Mode Bulk Out in Slave Mode

Figure 2-54 shows Host mode bulk OUT transfers in Slave mode. This example shows a 1-byte bulk OUT packet.

**Figure 2-54 Host Bulk OUT Transaction Using Slave Mode**



## 2.6 USB 1.1 FS Transceiver Interface

Figure 2-55 shows the USB 1.1 FS transceiver transmitting data.

**Figure 2-55 USB 1.1 FS Transceiver Transmitting Data**

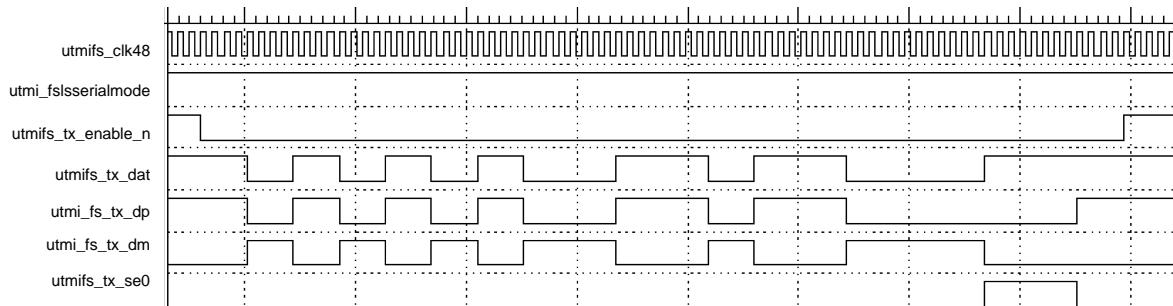
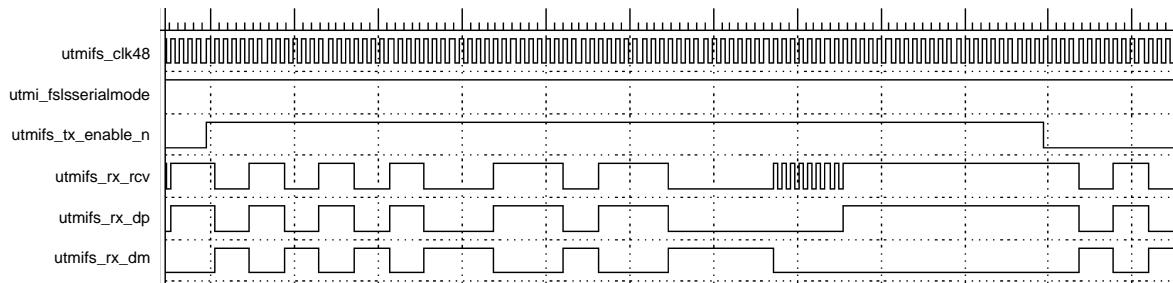


Figure 2-56 shows the USB 1.1 FS transceiver receiving data.

**Figure 2-56 USB 1.1 FS Transceiver Receiving Data**



## 2.7 Embedded Host with Micro-AB Receptacle Support

If you want to implement Embedded Host with micro-AB receptacle support, then you must set OTG\_MODE=0,1, or 5.

In non-SRP capable configurations, external logic is needed to control PHY DRVVBUS based on ID value because the OTG state machine is not present inside the controller and PHY DRVVBUS is not connected from the DWC\_otg controller. In such a configuration to support micro-AB receptacle, DRVVBUS must be asserted when ID corresponds to Host and de-asserted when ID corresponds to Device.

To enable micro-AB receptacle support when the controller is acting as an Embedded Host the utmiotg\_iddig input signal is present in SRP Capable Host only mode (OTG\_MODE=5) as well. If the Embedded host supports only Standard A Connector, hard wire utmiotg\_iddig input signal of DWC\_otg controller to zero.



# 3

## Configuration Parameters

---



**Attention** From the v4.20a release onwards, a few Controller Configuration options are no longer supported. For more details, see [Note](#) on page 36.

---

This chapter details all the configuration parameters. You can use the coreConsultant GUI configuration reports to determine the actual configured state of the controller. Some expressions might refer to TCL functions or procedures (sometimes identified as <functionof>) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the controller in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

The parameter descriptions in this chapter include the **Enabled:** attribute which indicates the values required to be set on other parameters before you can change the value of this parameter.

These tables define all of the configuration options for this component.

- Basic Config on [page 142](#)
- USB Physical Layer Interface on [page 146](#)
- Device Endpoint Configuration on [page 150](#)
- Host Endpoint Configuration on [page 152](#)
- Endpoint Channel FIFO Configuration on [page 153](#)
- Additional Configuration Options on [page 158](#)
- Endpoint Direction on [page 164](#)
- Device Periodic FIFO Depth on [page 165](#)
- Device IN Endpoint FIFO Depth on [page 166](#)

## 3.1 Basic Config Parameters

**Table 3-1 Basic Config Parameters**

Label	Description
Mode of Operation	<p>Selects the mode of operation for DWC_otg. Host-only and Device-only modes reduce area by 2-3K gates. The Non-HNP and Non-SRP modes reduce area by 0.5K gates each. OTG, HNP, and SRP capabilities selected using this parameter can be disabled by software after power-up.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>■ The Non-HNP/SRP-Capable OTG option lacks dynamic switching capability, but can work either as a host or a device depending on whether it is attached to an "A" or "B" connector: <ul style="list-style-type: none"> <li>- Host: When attached to an A connector.</li> <li>- Device: When attached to a B connector.</li> </ul> </li> <li>■ If you want to implement Embedded Host with micro-AB receptacle support, then you must set OTG_MODE=0,1, or 5.</li> </ul> <p><b>Dependencies:</b></p> <p>For HNP and SRP configurations when parameter OTG_FSPHY_INTERFACE=1, parameter OTG_I2C_INTERFACE must not be 0 if your FS dedicated PHY supports HNP and SRP through I2C register programming.</p> <p>Valid licenses are required for the selected mode.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ RESERVED-0 (0)</li> <li>■ RESERVED-1 (1)</li> <li>■ Non-HNP-Non-SRP-Capable Device &amp; Host (2)</li> <li>■ RESERVED-3 (3)</li> <li>■ Non-OTG Device (4)</li> <li>■ RESERVED-5 (5)</li> <li>■ Non-OTG Host (6)</li> </ul> <p><b>Default Value:</b> Non-HNP-Non-SRP-Capable Device &amp; Host</p> <p><b>Enabled:</b> Always</p> <p><b>Parameter Name:</b> OTG_MODE</p>

**Table 3-1 Basic Config Parameters (Continued)**

Label	Description
Architecture	<p>Specifies the architecture for the DWC_otg core.</p> <ul style="list-style-type: none"> <li>■ Choose Slave-Only mode when area is a concern and there is enough CPU bandwidth to execute the USB driver to move data between memory and the DWC_otg.</li> <li>■ Choose Internal DMA mode when CPU bandwidth to process USB transfers is limited, and the internal DMA controller will handle moving data between memory and the DWC_otg core.</li> <li>■ Choose External DMA mode if there is already a DMA controller on the SoC that has additional DMA channels for the DWC_otg core. Slave hardware is instantiated even if a DMA mode is selected. After power-up, software can disable DMA to use Slave-Only mode and vice-versa.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ Slave-Only (0)</li> <li>■ RESERVED-1 (1)</li> <li>■ Internal DMA (2)</li> </ul> <p><b>Default Value:</b> Slave-Only</p> <p><b>Enabled:</b> Always</p> <p><b>Parameter Name:</b> OTG_ARCHITECTURE</p>
Point-to-Point Application Only?	<p>Specifies whether the DWC_otg core will be used only in a point-to-point application or in a multi-point application. In a point-to-point application, the DWC_otg does not support hubs or splits in Host mode; it can only be directly connected to a HS, FS, or LS device. For each host channel, about 0.25K of gates are saved in point-to-point mode.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> No</p> <p><b>Enabled:</b> OTG_MODE!=3 &amp;&amp; OTG_MODE!=4 &amp;&amp; PHY_INST==0</p> <p><b>Parameter Name:</b> OTG_SINGLE_POINT</p>

**Table 3-1 Basic Config Parameters (Continued)**

Label	Description
LPM Mode of Operation	<p>Selects the Link Power Management (LPM) mode of operation for DWC_otg. The LPM increases area in the following modes:</p> <ul style="list-style-type: none"> <li>■ Device mode: increases area by approx 1.2K gates</li> <li>■ Host mode: increases area by approx 1.2K gates</li> <li>■ OTG mode: increases area by approx 2.2K gates</li> </ul> <p>LPM capabilities selected using this parameter can be disabled by the application after power-up. This feature requires access to the LPM Add-On License.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ Non-LPM Capable Core (0)</li> <li>■ LPM capable Core (1)</li> </ul> <p><b>Default Value:</b> Non-LPM Capable Core</p> <p><b>Enabled:</b> Always</p> <p><b>Parameter Name:</b> OTG_ENABLE_LPM</p>
Enable descriptor based scatter/gather DMA	<p>Enables descriptor-based scatter/gather DMA.</p> <ul style="list-style-type: none"> <li>■ When this parameter is enabled, DMA operations will be serviced with descriptor-based scatter/gather DMA. This option is available only with internal DMA mode.</li> <li>■ When this parameter is disabled, the DWC_otg controller is backward compatible to version 2.66a with buffer pointer based DMA implementation. This option is not available in Device mode when single transmit FIFO configuration is selected.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> No</p> <p><b>Enabled:</b> OTG_ARCHITECTURE==2</p> <p><b>Parameter Name:</b> OTG_EN_DESC_DMA</p>
Enable Dedicated Transmit FIFO for device IN Endpoints?	<p>Specifies whether Dedicated Transmit FIFOs should be enabled in device mode. If chosen, a separate Tx FIFO will be used for each of the device mode IN endpoints. There will not be dedicated transmit FIFOs for Host mode Tx channels. Choosing this option will cost additional 2.5K per IN endpoint, and 5K gates globally.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ RESERVED-0 (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> ((OTG_EN_DESC_DMA==1 &amp;&amp; OTG_MODE!=5 &amp;&amp; OTG_MODE!=6) ? 1 : 0)</p> <p><b>Enabled:</b> OTG_MODE!=5 &amp;&amp; OTG_MODE!=6 &amp;&amp; OTG_ARCHITECTURE!=1 &amp;&amp; OTG_EN_DESC_DMA==0</p> <p><b>Parameter Name:</b> OTG_EN_DED_TX_FIFO</p>

**Table 3-1 Basic Config Parameters (Continued)**

Label	Description
Enable option for endpoint specific interrupt	<p>Enables the Multi Processor Interrupt Functionality. This parameter enables the endpoint-specific interrupt pins and the endpoint specific mask registers.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> No</p> <p><b>Enabled:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6</p> <p><b>Parameter Name:</b> OTG_MULTI_PROC_INTRPT</p>

## 3.2 USB Physical Layer Interface Parameters

**Table 3-2 USB Physical Layer Interface Parameters**

Label	Description
<b>High-Speed PHY Interface(s)</b>	<p>Specifies the High-Speed PHY interface(s). Choose both UTMI+ and ULPI if you are not sure whether a UTMI+ or ULPI off-chip PHY will be used in the product, or if you have an on-chip UTMI+ PHY and want to bring out the ULPI interface as a backup in case the on-chip PHY fails. Because the MAC connects to the UTMI+ interface, choosing a ULPI interface adds about 1.5K gates for the ULPI-to-UTMI+ conversion logic. If the UTMI+ and ULPI option is chosen, software can select either interface.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ None (0)</li> <li>■ UTMI+ (1)</li> <li>■ ULPI (2)</li> <li>■ UTMI+ and ULPI (3)</li> </ul> <p><b>Default Value:</b> UTMI+</p> <p><b>Enabled:</b> Always</p> <p><b>Parameter Name:</b> OTG_HSPHY_INTERFACE</p>
<b>Data Width of the UTMI+ Interface</b>	<p>Specifies the data width of the UTMI+ PHY interface. Choose either 8 bits or 16 bits if your UTMI+ PHY only supports one of these. If you are not sure which PHY you will be using in your product, choose 8/16 bits (software selectable). Choosing 8 bits or 16 bits only reduces area by 0.5K gates. Running the MAC UTMI+ interface at 16 bits will consume less power because the MAC runs at 30 MHz instead of 60 MHz. When a ULPI PHY is used, an internal wrapper converts ULPI to UTMI+. Because the ULPI-to-UTMI+ wrapper adds additional delay when ULPI PHY is selected, only 8-bit mode should be used.</p> <p>To meet USB turnaround time, select either a hardware-fixed 8-bit mode using this parameter, or 8-bit mode using software when 8/16 bit mode is selected for hardware.</p> <p>If the controller is operating in device mode and is required to support multiple layers of hubs, then to meet USB turnaround time, select either a hardware-fixed 8-bit mode using this parameter, or 8-bit mode using software when 8/16 bit mode is selected for hardware. In addition to configuring/programming the controller to operate in 8-bit mode, the AHB frequency must also be selected based on the number of layers of hubs that the USB device must support</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 8 bits (0)</li> <li>■ 16 bits (1)</li> <li>■ 8/16 bits (2)</li> </ul> <p><b>Default Value:</b> 8/16 bits</p> <p><b>Enabled:</b> OTG_HSPHY_INTERFACE!=0</p> <p><b>Parameter Name:</b> OTG_HSPHY_DWIDTH</p>

**Table 3-2 USB Physical Layer Interface Parameters (Continued)**

Label	Description
USB 1.1 Full-Speed Serial Transceiver Interface	<p>Specifies the USB 1.1 Full-Speed Serial Transceiver interface. You can choose dedicated USB 1.1 FS pins, or the pins can be shared with a UTMI+ or ULPI interface. Implement both a FS transceiver interface and a HS PHY interface if you want to use the same chip in FS and HS products, or have the full-speed interface as a backup in case the on-chip high-speed PHY fails. Selecting the Dedicated FS interface adds about 3.5K gates.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ None (0)</li> <li>■ Dedicated FS (1)</li> <li>■ FS pins shared with UTMI+ (2)</li> <li>■ FS pins shared with ULPI (3)</li> </ul> <p><b>Default Value:</b> None</p> <p><b>Enabled:</b> Always</p> <p><b>Parameter Name:</b> OTG_FSPHY_INTERFACE</p>
USB IC_USB Transceiver Interface	<p>Specifies the IC_USB Full-Speed Serial Transceiver interface. You can choose dedicated USB 1.1 FS pins, or the IC_USB additional interface. Implement both a FS transceiver interface and a IC_USB transceiver interface if you want to use the same chip in FS and IC_USB products. Selecting the IC_USB interface adds about 400 gates.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ None (0)</li> <li>■ IC_USB interface (1)</li> </ul> <p><b>Default Value:</b> None</p> <p><b>Enabled:</b> OTG_FSPHY_INTERFACE!=0</p> <p><b>Parameter Name:</b> OTG_ENABLE_IC_USB</p>
Default (Power on) Interface selection: FS_USB/IC_USB	<p>Specifies the initial mode of controller when IC_USB is enabled.</p> <ul style="list-style-type: none"> <li>■ Select IC_USB interface if controller needs to come up with IC_USB mode as default.</li> <li>■ Select FS_USB interface if controller needs to come up with FS_USB mode as default</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ FS_USB interface (0)</li> <li>■ IC_USB interface (1)</li> </ul> <p><b>Default Value:</b> FS_USB interface</p> <p><b>Enabled:</b> OTG_ENABLE_IC_USB!=0</p> <p><b>Parameter Name:</b> OTG_SELECT_IC_USB</p>

**Table 3-2 USB Physical Layer Interface Parameters (Continued)**

Label	Description
HSIC Mode of Operation	<p>Selects the High Speed Interchip (HSIC) mode of operation for DWC_otg. The HSIC feature increases area in the following modes:</p> <ul style="list-style-type: none"> <li>■ Device mode: increases area by approx 175 gates</li> <li>■ Host mode: increases area by approx 75 gates</li> <li>■ OTG mode: increases area by approx 235 gates</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ Non-HSIC Capable Core (0)</li> <li>■ HSIC capable Core (1)</li> </ul> <p><b>Default Value:</b> Non-HSIC Capable Core</p> <p><b>Enabled:</b> (OTG_HSPHY_INTERFACE==1    OTG_HSPHY_INTERFACE==3) &amp;&amp; OTG_EN_PWROPT!=3</p> <p><b>Parameter Name:</b> OTG_ENABLE_HSIC</p>
Enable I2C Interface?	<p>Specifies whether the I2C interface is enabled. The I2C interface can be used for OTG control instead of, or in addition to, the regular OTG interface on the USB 1.1 Full-Speed OTG Transceiver (UTMIFS).</p> <p>Using the I2C interface reduces the OTG pin count from 11 to 3. The I2C interface is also used for CarKit applications in OTG and non-OTG configurations. This feature adds about 1.5K gates.</p> <ul style="list-style-type: none"> <li>■ For OTG mode function (HNP and SRP with FS transceiver Interface, the I2C interface must be enabled to control OTG control signals.</li> <li>■ If carKit mode is used with FS transceiver Interface, then I2C interface must be enabled to access carKit registers.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ I2C Only (1)</li> <li>■ I2C and UTMIFS (2)</li> </ul> <p><b>Default Value:</b> No</p> <p><b>Enabled:</b> OTG_FSPHY_INTERFACE!=0</p> <p><b>Parameter Name:</b> OTG_I2C_INTERFACE</p>
Enable ULPI Carkit?	<p>Specifies whether the Carkit is enabled for interface. If enabled, the software uses the ULPI interface for ULPI PHY internal register access. Software can access ULPI PHY internal registers using register read/writes to DWC_otg.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> No</p> <p><b>Enabled:</b> OTG_HSPHY_INTERFACE==2    OTG_HSPHY_INTERFACE==3</p> <p><b>Parameter Name:</b> OTG_ULPI_CARKIT</p>

**Table 3-2 USB Physical Layer Interface Parameters (Continued)**

Label	Description
ADP Controller Support	<p>This parameter determines the nature of ADP controller support needed.</p> <ul style="list-style-type: none"> <li>■ No: Choosing the parameter to this value will make the HSOTG core support Option1 mode of ADP support.</li> <li>■ Yes: Choosing the parameter to this value will infer ADP control logic outside the HSOTG core (Option2 mode of ADP support).</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> No</p> <p><b>Enabled:</b> OTG_MODE!=2 &amp;&amp; OTG_MODE!=4 &amp;&amp; OTG_MODE!=6 &amp;&amp; (OTG_HSPHY_INTERFACE!=0    OTG_I2C_INTERFACE!=1)</p> <p><b>Parameter Name:</b> OTG_ADP_SUPPORT</p>
Battery Charger Support	<p>This parameter determines whether support for Battery Charger is required.</p> <ul style="list-style-type: none"> <li>■ No: Choosing the parameter to this value will make the HSOTG core not provide BC pins.</li> <li>■ Yes: Choosing the parameter to this value will make the HSOTG core provide BC pins.</li> </ul> <p>Note: The Battery Charger Controller is not implemented in the Core. The core only provides a Register interface to the Rid inputs and also detects changes to those inputs. Please review the BC related sections of the Databook</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> No</p> <p><b>Enabled:</b> Always</p> <p><b>Parameter Name:</b> OTG_BC_SUPPORT</p>
Enable PHY Vendor Control Interface?	<p>Specifies whether the Vendor Control interface is enabled. If enabled, ULPI/UTMI+ PHY internal registers can be accessed by software using register read/writes to DWC_otg. For ULPI PHYs, internal registers are accessed through the same ULPI interface used for USB transfers (arbitrated between USB and software access). UTMI+ PHYs have a separate Vendor Control interface. This feature adds about 0.7K gates.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> ((PHY_INST==1) ? 1 : 0)</p> <p><b>Enabled:</b> OTG_HSPHY_INTERFACE!=0 &amp;&amp; PHY_INST==0</p> <p><b>Parameter Name:</b> OTG_VENDOR_CTL_INTERFACE</p>

### 3.3 Device Endpoint Configuration Parameters

**Table 3-3 Device Endpoint Configuration Parameters**

Label	Description
Number of Device Mode Endpoints in Addition to Control Endpoint 0	<p>Specifies the number of Device mode endpoints in addition to endpoint 0, which is always present. Choose the maximum possible number of endpoints that must be supported for all configurations and alternate settings. Specifying additional endpoints is not useful because the core will not use the excess registers, resulting in increased gate count but no gain in performance.</p> <p>If area is not a concern, and you want a flexible design that can be used in multiple applications, instantiate all 15 endpoints.</p> <p><b>Values:</b> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15  <b>Default Value:</b> 2  <b>Enabled:</b> OTG_MODE!=5 &amp;&amp; OTG_MODE!=6  <b>Parameter Name:</b> OTG_NUM_EPS</p>
Number of Device Mode Periodic IN Endpoints	<p>Specifies the maximum number of Device mode IN Endpoints that can support periodic transfers.</p> <p>Each periodic IN endpoint needs a separate Periodic TxFIFO Controller that is 0.6K gates. In addition, 1 maximum packet size (MPS) of memory must be reserved in the Data FIFO.</p> <p>For high-bandwidth endpoints, the FIFO memory requirement is 2 KB or 3 KB, depending on whether two or three transactions are sent per microframe. If this parameter is set to 0, no Device Periodic TxFIFOs will be instantiated.</p> <p><b>Values:</b> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15  <b>Default Value:</b> 0  <b>Enabled:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 &amp;&amp; OTG_EN_DED_TX_FIFO ==0  <b>Parameter Name:</b> OTG_NUM_PERIO_EPS</p>
Number of Device Mode active IN Endpoints Including Control Endpoint 0	<p>Specifies the maximum number of Device mode IN endpoints active at any time including endpoint 0, which is always present. This parameter determines the number of Device mode Tx FIFOs to be instantiated.</p> <p><b>Values:</b> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16  <b>Default Value:</b> 3  <b>Enabled:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 &amp;&amp; OTG_EN_DED_TX_FIFO ==1  <b>Parameter Name:</b> OTG_NUM_IN_EPS</p>

**Table 3-3 Device Endpoint Configuration Parameters (Continued)**

Label	Description
Number of Device Mode Control Endpoints in Addition to Endpoint 0	<p>Specifies the number of Device mode control endpoints in addition to control endpoint 0, which is always present.</p> <p>This parameter is used to reserve four 32-bit locations for each control endpoint in the RxFIFO. SETUP transactions cannot be NAKed, guaranteeing that one SETUP transaction for each control endpoint can be received at any time.</p> <p>To handle back-to-back SETUP transactions to a given endpoint, six additional 32-bit locations, common to all control endpoints, are reserved to accept two more SETUP transactions for any control endpoint.</p> <p><b>Values:</b> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15  <b>Default Value:</b> 0  <b>Enabled:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6  <b>Parameter Name:</b> OTG_NUM_CRL_EPS</p>

## 3.4 Host Endpoint Configuration Parameters

**Table 3-4 Host Endpoint Configuration Parameters**

Label	Description
Number of Host Mode Channels	<p>Specifies the number of Host mode channels. Because Device mode IN and OUT endpoint registers are reused as host registers in Host mode, to save area, the number of host channels should be the same as the total number of Device IN and OUT endpoints.</p> <p>For example, if you have one IN and OUT endpoint, three IN endpoints, and five OUT endpoints, then instantiating up to 10 (<math>1 * 2 + 3 + 5</math>) host channels will not add additional area overhead. For data transfers, each host channel is mapped to a device endpoint.</p> <p>In Host mode, if the number of device endpoints to be supported is more than the number of host channels, software can reprogram the channels to support up to 127 devices, each having 32 endpoints (IN + OUT), for a maximum of 4,064 endpoints.</p> <p>The number of host channels determines the width of DMA channels:</p> <ul style="list-style-type: none"> <li>■ 4 host channels --&gt; [2:0]. bit 2: in/out, bit [1:0] channel number</li> <li>■ 8 host channels --&gt; [3:0]. bit 3: in/out, bit [2:0] channel number</li> <li>■ 16 host channels --&gt; [4:0]. bit 4: in/out, bit [3:0] channel number</li> </ul> <p><b>Values:</b> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16</p> <p><b>Default Value:</b> 4</p> <p><b>Enabled:</b> OTG_MODE != 3 &amp;&amp; OTG_MODE != 4</p> <p><b>Parameter Name:</b> OTG_NUM_HOST_CHAN</p>
Is Periodic OUT Channel Support Needed in Host Mode?	<p>If Host mode periodic OUT support is needed, a Periodic Tx FIFO Controller is instantiated in the design. In addition, a minimum of one maximum packet size (MPS) of FIFO memory must be reserved for host periodic transfers. This FIFO is shared with Device Periodic Tx FIFO 1 in shared FIFO mode and Device IN endpoint Tx FIFO 1 in dedicated FIFO mode.</p> <p>For example, in shared FIFO mode (OTG_ENDED_TX_FIFO==0), if you need Host mode periodic transfer support and periodic support for 4 Device mode endpoints, one 2K-gate host Periodic Tx FIFO Controller and three 0.6K-gate device Periodic Tx FIFO Controllers will be instantiated. Device Periodic FIFO controllers are smaller than the host FIFO controller because they hold only one packet at a time.</p> <p>In dedicated FIFO mode (OTG_ENDED_TX_FIFO==1), if you need host periodic support, the host periodic FIFO will be shared with device endpoint FIFO 1 (2K gate).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> ((PHY_INST==0) ? 0 : 1)</p> <p><b>Enabled:</b> OTG_MODE != 3 &amp;&amp; OTG_MODE != 4 &amp;&amp; PHY_INST==0</p> <p><b>Parameter Name:</b> OTG_EN_PERIO_HOST</p>

## 3.5 Endpoint Channel FIFO Configuration Parameters

**Table 3-5 Endpoint Channel FIFO Configuration Parameters**

Label	Description
Total Data FIFO RAM Depth (64 to 32768)	<p>Specifies the depth of the combined Rx and Tx Data FIFOs. The FIFO RAM width is 35 bits: 32 data bits plus 3 control bits. This is the greater of the Host mode or Device mode FIFO memory requirement.</p> <ul style="list-style-type: none"> <li>■ For Host mode, the memory requirement is the sum of: RxFIFO + Non-Periodic TxFIFO + Host Periodic TxFIFO + Channel specific DMA address space (one location per channel for Buffer DMA, and four locations per channel for Scatter/Gather DMA)</li> <li>■ In Device mode, the memory requirement is the sum of: RxFIFO + Non-Periodic TxFIFO + Device Periodic TxFIFOs + Endpoint specific DMA address space (one location per endpoint for Buffer DMA, and four locations per endpoint for Scatter/Gather DMA)</li> <li>■ If dedicated Tx FIFO is not enabled, it is the sum of: RxFIFO + Sum of all In endpoint TxFIFOs.</li> </ul> <p><b>Values:</b> 64, ..., 32768  <b>Default Value:</b> 1024  <b>Enabled:</b> Always  <b>Parameter Name:</b> OTG_DFIFO_DEPTH</p>
Enable Dynamic FIFO Sizing?	<p>Specifies whether dynamic FIFO sizing is enabled. If enabled, the FIFO depths can be changed by software, depending on the endpoint configuration. Supporting dynamic FIFO sizing increases design flexibility for different applications by allowing software to reallocate memory. However, if your USB configuration does not change, disabling dynamic FIFO sizing optimizes the FIFO start point and size registers to save area.</p> <p><b>Note:</b> Device Periodic TxFIFO depths are changed by programming their respective start addresses (Valid only in shared FIFO mode OTG_EN_DED_TX_FIFO==0).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ RESERVED-0 (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> Yes  <b>Enabled:</b> PHY_INST==0  <b>Parameter Name:</b> OTG_DFIFO_DYNAMIC</p>

**Table 3-5 Endpoint Channel FIFO Configuration Parameters (Continued)**

Label	Description
Largest Rx Data FIFO Depth (16 to 32768)	<p>Specifies the largest RxFIFO depth and the memory allocation for the host/device RxFIFO when dynamic FIFO sizing is disabled.</p> <p>If dynamic FIFO sizing is enabled, software can set the RxFIFO size up to the value specified for this parameter. This parameter controls the RxFIFO Pointer size. It is recommended to set this to Total Data FIFO RAM Depth, to provide a flexible FIFO architecture. The RxFIFO holds received packets and status information for the packets.</p> <p>Please go through the Programming Guide for details on selecting the FIFO Size. The minimum depth is as follows:</p> <p><b>Buffer DMA or Slave DMA Mode:</b></p> $(5 * \text{number of control endpoints} + 8 + (1 + \text{largest USB packet size in bytes}/4)) + 1 + 2 * \text{Total Number of OUT Endpoints}$ <p>The recommended depth is: <math>(5 * \text{number of control endpoints} + 8 + (2 * (1 + \text{largest USB packet size in bytes}/4))) + 1 + (2 * \text{Number of OUT Endpoints})</math></p> <p><b>Scatter/Gather DMA Mode:</b></p> $(4 * \text{number of control endpoints} + 6 + (1 + \text{largest USB packet size in bytes}/4)) + 1 + 2 * \text{Total Number of OUT Endpoints}$ <p>The recommended depth is: <math>(4 * \text{number of control endpoints} + 6 + (2 * (1 + \text{largest USB packet size in bytes}/4))) + 1 + (2 * \text{Number of OUT Endpoints})</math></p> <p>Although 1 maximum packet size (MPS) of memory is sufficient, allotting twice this amount allows simultaneous operation. That is, when the USB is writing one packet, the AHB can read the previous packet.</p> <p>In device mode, if high- bandwidth isochronous support is needed, or more than one isochronous endpoint is present, then 2 MPS of memory is the minimum required to hold back-to-back packets. If the AHB latency is high or the AHB clock is slow, allot more than 2 MPS to hold multiple received packets. Along with each packet, a 32-bit block of status information is stored in this FIFO. In addition, one more status information also pushed in to the fifo at the end of each transfer. One additional FIFO location is needed for Global NAK operation.</p> <p>For example in Scatter/Gather DMA mode, if the largest packet is 1,024 bytes, and there is one control endpoint and two OUT endpoints, the recommended memory allocation is:</p> $(4 * 1 + 6) + (2 * (1 + 1,024 / 4)) + 2 * 2 + 1 = 527$ <p>If you are planning to support thresholding in Scatter/Gather DMA mode, the minimum RxFIFO requirement is <math>(4 * \text{number of Control endpoints} + 6) + 2 * ((\text{Rx_threshold size}/4) + 1) + 1</math> for last threshold + 1 for transfer complete + 1 for Global OUT NAK.</p> <p>The RxFIFO is a shared FIFO between endpoints or channels in the Device or Host mode, respectively. All endpoints or channels share the FIFO but they do not partition it.</p> <p><b>Values:</b> 16, ..., 32768</p> <p><b>Default Value:</b> 1024</p> <p><b>Enabled:</b> Always</p> <p><b>Parameter Name:</b> OTG_RX_DFIFO_DEPTH</p>

**Table 3-5 Endpoint Channel FIFO Configuration Parameters (Continued)**

Label	Description
Largest Non-Periodic Host Tx Data FIFO Depth (16 to 32768)	<p>Specifies the largest Non-Periodic TxFIFO depth and the memory allocation for the host Non-Periodic TxFIFO when dynamic FIFO sizing is disabled. If dynamic FIFO sizing is enabled, software can set the host Non-Periodic FIFO size up to the value specified for this parameter. This parameter defines the Non-Periodic TxFIFO pointer size. It is recommended to set this to Total Data // FIFO RAM Depth, to provide a flexible FIFO architecture.</p> <p>Please go through the Programming Guide for details on selecting the FIFO Size.</p> <ul style="list-style-type: none"> <li>■ The minimum depth is (largest Non-Periodic Transmit USB packet size in bytes/4).</li> <li>■ The recommended depth is (2 * largest Non-Periodic Transmit USB packet size in bytes/4).</li> </ul> <p>Although 1 maximum packet size (MPS) of memory is sufficient, allotting twice this amount allows simultaneous operation. That is, when the USB is reading one packet, the AHB can write the next packet. If the AHB latency is high or the AHB clock is slow, allot more than 2 MPS of memory to hold multiple non-periodic transmit packets. Because the largest non-periodic packet size is 512 bytes, the recommended FIFO depth is <math>2 * (512/4) = 256</math>. The host mode non periodic FIFO will be shared with Device mode IN endpoint FIFO #0.</p> <p><b>Values:</b> 16, ..., 32768  <b>Default Value:</b> 1024  <b>Enabled:</b> OTG_MODE != 3 &amp;&amp; OTG_MODE != 4  <b>Parameter Name:</b> OTG_TX_HNPERIO_DFIFO_DEPTH</p>
Largest Non-Periodic Tx Data FIFO Depth (16 to 32768)	<p>Specifies the largest Non-Periodic TxFIFO depth and the memory allocation for the host/device Non-Periodic TxFIFO when dynamic FIFO sizing is disabled. If dynamic FIFO sizing is enabled, software can set the Non-Periodic FIFO size up to the value specified for this parameter. This parameter defines the Non-Periodic TxFIFO pointer size. It is recommended to set this to Total Data FIFO RAM Depth, to provide a flexible FIFO architecture.</p> <p>Please go through the Programming Guide for details on selecting the FIFO Size.</p> <ul style="list-style-type: none"> <li>■ The minimum depth is (largest Non-Periodic Transmit USB packet size in bytes/4).</li> <li>■ The recommended depth is (2 * largest Non-Periodic Transmit USB packet size in bytes/4).</li> </ul> <p>Although 1 maximum packet size (MPS) of memory is sufficient, allotting twice this amount allows simultaneous operation. That is, when the USB is reading one packet, the AHB can write the next packet. If the AHB latency is high or the AHB clock is slow, allot more than 2 MPS of memory to hold multiple non-periodic transmit packets.</p> <p>Because the largest non-periodic packet size is 512 bytes, the recommended FIFO depth is <math>2 * (512/4) = 256</math>.</p> <p><b>Values:</b> 16, ..., 32768  <b>Default Value:</b> 1024  <b>Enabled:</b> OTG_ENDED_TxFIFO==0  <b>Parameter Name:</b> OTG_TX_NPERIO_DFIFO_DEPTH</p>

**Table 3-5 Endpoint Channel FIFO Configuration Parameters (Continued)**

Label	Description
Largest Host Mode Periodic Tx Data FIFO Depth (16 to 32768)	<p>Specifies the largest Host mode Periodic TxFIFO depth and the memory allocation for the Periodic TxFIFO when dynamic FIFO sizing is disabled. If dynamic FIFO sizing is enabled, software can set the host Periodic TxFIFO size up to the value specified for this parameter. This parameter also defines the Host Periodic TxFIFO pointer size.</p> <ul style="list-style-type: none"> <li>■ The minimum depth is (largest periodic transmit USB packet size in bytes/4).</li> <li>■ The recommended depth is (2 * largest periodic transmit USB packet size in bytes/4)).</li> </ul> <p>Please go through the Programming Guide for details on selecting the FIFO Size. Although one maximum packet size (MPS) of memory is sufficient, allotting twice this amount allows simultaneous operation. That is, when the USB is reading one packet, the AHB can write the next packet. If high-bandwidth isochronous support is needed, or more than one isochronous endpoint is present, 2 MPS of memory is the minimum required to hold back-to-back packets. If the AHB latency is high or the AHB clock is slow, allot more than 2 MPS of memory to hold multiple periodic transmit packets. Because the largest periodic packet size is 1,024 bytes, the recommended depth is FIFO depth = 2 * (1,024/4) = 512.</p> <p><b>Values:</b> 16, ..., 32768  <b>Default Value:</b> 1024  <b>Enabled:</b> OTG_MODE != 3 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_EN_PERIO_HOST==1  <b>Parameter Name:</b> OTG_TX_HPERIO_DFIFO_DEPTH</p>
Non-Periodic Request Queue Depth	<p>Specifies the Non-Periodic Request Queue depth. That is, the maximum number of packets that can reside in the Non-Periodic TxFIFO.</p> <ul style="list-style-type: none"> <li>■ In Device mode, the queue is used only in Shared FIFO Mode (Enable Dedicated Transmit FIFO for device IN Endpoints? ==No).</li> <li>■ In Device mode shared FIFO mode, there is one entry in the Non-Periodic Request Queue for each packet in the Non-Periodic TxFIFO.</li> <li>■ In Host mode, this queue holds one entry corresponding to each IN or OUT non-periodic request. This queue is 7 bits wide.</li> </ul> <p><b>Values:</b> 2, 4, 8  <b>Default Value:</b> ((PHY_INST==0) ? 8 : 8)  <b>Enabled:</b> PHY_INST==0 &amp;&amp; (!((OTG_MODE == 3    OTG_MODE == 4) &amp;&amp; OTG_ENDED_TX_FIFO==1))  <b>Parameter Name:</b> OTG_NPERIO_TX_QUEUE_DEPTH</p>
Host Mode Periodic Request Queue Depth	<p>Specifies the Host mode Periodic Request Queue depth. That is, the maximum number of packets that can reside in the Host Periodic TxFIFO. This queue holds one entry corresponding to each IN or OUT periodic request. This queue is 9 bits wide.</p> <p><b>Values:</b> 2, 4, 8, 16  <b>Default Value:</b> ((PHY_INST==0) ? 8 : 8)  <b>Enabled:</b> OTG_MODE!=3 &amp;&amp; OTG_MODE!=4 &amp;&amp; OTG_EN_PERIO_HOST==1 &amp;&amp; PHY_INST==0  <b>Parameter Name:</b> OTG_PERIO_TX_QUEUE_DEPTH</p>

**Table 3-5 Endpoint Channel FIFO Configuration Parameters (Continued)**

Label	Description
Device Mode IN Token Sequence Learning Queue Depth (0 to 30)	<p>Specifies the Device mode IN Token Sequence Learning Queue depth. This queue logs the endpoint numbers of non-periodic IN tokens. It is used by software to set up packets in the Non-Periodic TxFIFO in the order the host will request them. If disabled, this queue is not instantiated. When there are only two Tx endpoints, this queue is usually not needed. This queue is 4 bits wide.</p> <p><b>Values:</b> 0, ..., 30  <b>Default Value:</b> 8  <b>Enabled:</b> OTG_MODE!=5 &amp;&amp; OTG_MODE!=6 &amp;&amp; OTG_EN_DED_TX_FIFO==0 &amp;&amp; PHY_INST==0  <b>Parameter Name:</b> OTG_TOKEN_QUEUE_DEPTH</p>

## 3.6 Additional Configuration Options Parameters

**Table 3-6 Additional Configuration Options Parameters**

Label	Description
<b>Enable Service Interval Based-Scheduling for ISOC IN Endpoints</b>	<p>Enables service interval based scheduling for ISOC IN endpoints. When this feature is enabled, the Device Controller holds ISOC IN data for a given isochronous service interval in the TxFIFO until the end of the last frame/microframe of the service interval for that ISOC IN endpoint. If the USB Host sends ISOC IN token(s) in any micro frame of the service interval, the Device Controller responds with ISOC IN data corresponding to that service interval.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>■ This feature is applicable only to Device Descriptor DMA mode of operation for non-OTG device and non-OTG DRD V4 controller configurations.</li> <li>■ When this configuration option is enabled, the Controller uses a free-running external reference clock (ref_clk) which must be provided by the SoC/system to keep track of the frame/microframe counters. This allows the Device Controller to fetch and flush the data during L1 to support ADC 3.0 requirements. For additional details regarding the reference clock input and its frequencies, refer to section "System Clock Speeds" in the databook.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ Disable (0)</li> <li>■ Enable (1)</li> </ul> <p><b>Default Value:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 &amp;&amp; OTG_EN_DESC_DMA == 1 &amp;&amp; OTG_MODE!=0 &amp;&amp; OTG_MODE!=1 &amp;&amp; OTG_MODE!=2 &amp;&amp; OTG_MODE!=3 &amp;&amp; &lt;DWC-OTG-V4 feature authorized&gt;==1</p> <p><b>Enabled:</b> OTG_EN_DESC_DMA==1 &amp;&amp; OTG_MODE!=5 &amp;&amp; OTG_MODE!=6 &amp;&amp; OTG_MODE!=0 &amp;&amp; OTG_MODE!=1 &amp;&amp; OTG_MODE!=2 &amp;&amp; OTG_MODE!=3 &amp;&amp; &lt;DWC-OTG-V4 feature authorized&gt;==1</p> <p><b>Parameter Name:</b> OTG_SERV_INT_ENH</p>
<b>Width of Transfer Size Counters</b>	<p>Specifies the width of the Transfer Size Counters. After power-up, this value can be changed by software. This value defines, in bytes, the largest transfer size supported by the core. A Transfer Count register field is associated with each device endpoint or host channel. For area-sensitive applications, use a smaller number.</p> <p><b>Important Note:</b> When Host Scatter/Gather DMA mode is selected, this parameter must be set greater than or equal to 16.</p> <p><b>Values:</b> 11, 12, 13, 14, 15, 16, 17, 18, 19</p> <p><b>Default Value:</b> ((PHY_INST==0) ? 19 : 19)</p> <p><b>Enabled:</b> PHY_INST==0</p> <p><b>Parameter Name:</b> OTG_TRANS_COUNT_WIDTH</p>

**Table 3-6 Additional Configuration Options Parameters (Continued)**

Label	Description
Width of Packet Counters	<p>Specifies the width of the Packet Counters. After power-up, this value can be changed by software. This value defines the maximum number of packets that can be sent/received by the core in a transfer. A Packet Count register field is associated with each device endpoint or host channel. For an area-sensitive application, use a smaller number.</p> <p><b>Values:</b> 4, 5, 6, 7, 8, 9, 10  <b>Default Value:</b> ((PHY_INST==0) ? 10 : 10)  <b>Enabled:</b> PHY_INST==0  <b>Parameter Name:</b> OTG_PACKET_COUNT_WIDTH</p>
Remove Optional Features?	<p>Specifies whether to remove optional features. When this parameter is enabled, the User ID register, General Purpose Input/Output ports, and SOF toggle and counter ports are removed to reduce area by 0.6K gates.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> ((PHY_INST==0) ? 1 : 0)  <b>Enabled:</b> PHY_INST==0  <b>Parameter Name:</b> OTG_RM_OPT_FEATURES</p>
Power-on Value of User ID Register	<p>Specifies the power-on value of the User ID register. After power-up, this value can be changed by software. This register can be used as either a scratch pad or an identification register.</p> <p><b>Values:</b> 0x0, ..., 0xffffffff  <b>Default Value:</b> 0x12345678  <b>Enabled:</b> OTG_RM_OPT_FEATURES==0  <b>Parameter Name:</b> OTG_USERID</p>
Enable Power Optimization?	<p>Specifies whether to enable power optimization. When enabled, clock gating and two power rail options are provided to gate the clocks off and switch off power to most of the DWC_otg core (except the Bus Interface Unit and PMU) during a suspend.</p> <p><b>Note:</b> Hibernation (2) and External Hibernation (3) options are available only if you have procured the 6922-0 DWC USB HSOTG Hibernate Add-On license.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ PartialPowerDown (1)</li> <li>■ Hibernation (2)</li> <li>■ RESERVED-3 (3)</li> </ul> <p><b>Default Value:</b> No  <b>Enabled:</b> Always  <b>Parameter Name:</b> OTG_EN_PWRLOPT</p>

**Table 3-6 Additional Configuration Options Parameters (Continued)**

Label	Description
Is Minimum AHB Operating Frequency less than 60 MHz?	<p>When the AHB frequency is less than 60 MHz, 4-deep clock-domain-crossing sink and source buffers are instantiated between the MAC and the Packet FIFO Controller (PFC); otherwise, 2-deep buffers are sufficient and save area.</p> <p>For Configurations using Dedicated TX FIFOs the 4-deep clock-domain-crossing sink and source buffers are always instantiated.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> Yes</p> <p><b>Enabled:</b> OTG_ENDED_TX_FIFO==0</p> <p><b>Parameter Name:</b> OTG_MIN_AHB_FREQ_LESS_THAN_60</p>
Reset Style for Clocked always Blocks in RTL?	<p>Specifies either asynchronous or synchronous reset coding style for all clocked always blocks in the RTL.</p> <p><b>Note:</b></p> <p>In synchronous reset style, the UTMI output signals from the controller will be X until the utmi_clk starts running. This is an expected behavior for synchronous reset configuration. Reset propagates to the controller only after the utmi_clk is running, so the state of all flops inside the controller will be undefined.</p> <p>If OTG_EN_PWR_OPT&lt;2 and OTG_AD_P SUPPORT=0, the utmi_opmode, utmi_termselect, and utmi_xcvrselect UTMI output signals are driven to PHY_NON_DRIVE, HS_TERM, and FS_XCVR values, respectively, even if the PHY clock is not present during power-on reset. If the PHY has an issue with UTMI signals being X until the utmi_clk starts, the PHY needs to be reset again.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ Asynchronous (0)</li> <li>■ Synchronous (1)</li> </ul> <p><b>Default Value:</b> Asynchronous</p> <p><b>Enabled:</b> Always</p> <p><b>Parameter Name:</b> OTG_SYNC_RESET_TYPE</p>

**Table 3-6 Additional Configuration Options Parameters (Continued)**

Label	Description
Enable Filter on "iddig" signal from PHY?	<p>Specifies whether to add a filter on the "iddig" input from the PHY. If your PHY already has a filter on iddig for de-bounce, then it is not necessary to enable the one in the DWC_otg. The filter is implemented in the DWC_otg_wpc module as a 20-bit counter that works on the PHY clock. In the case of the UTMI+ PHY, this pin is from the PHY. In the case of the ULPI PHY, this signal is generated by the ULPI Wrapper inside the core.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> (((OTG_MODE == 0    OTG_MODE == 1    OTG_MODE == 2) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1))) ? 1 : 0)</p> <p><b>Enabled:</b> ((OTG_MODE == 0    OTG_MODE == 1    OTG_MODE == 2) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Parameter Name:</b> OTG_EN_IDDIG_FILTER</p>
Enable Filter on "vbus_valid" signal from PHY?	<p>Specifies whether to add a filter on the "vbus_valid" input from the PHY. If your PHY already has a filter on vbus_valid, then it is not necessary to enable the one in the DWC_otg. The filter is implemented in the DWC_otg_wpc module as an 18-bit counter that works on the PHY clock. In the case of the UTMI+ PHY, this pin is from the PHY. In the case of the ULPI PHY, this signal is generated by the ULPI Wrapper inside the core.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> (((OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1))) ? 1 : 0)</p> <p><b>Enabled:</b> ((OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Parameter Name:</b> OTG_EN_VBUSVALID_FILTER</p>

**Table 3-6 Additional Configuration Options Parameters (Continued)**

Label	Description
Enable Filter on "a_valid" signal from PHY?	<p>Specifies whether to add a filter on the "a_valid" input from the PHY. If your PHY already has a filter on a_valid, then it is not necessary to enable the one in the DWC_otg. The filter is implemented in the DWC_otg_wpc module as an 8-bit counter that works on the PHY clock. In the case of the UTMI+ PHY, this pin is from the PHY. In the case of ULPI PHY, this signal is generated by the ULPI Wrapper inside the core.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> (((OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1))) ? 1 : 0)</p> <p><b>Enabled:</b> ((OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Parameter Name:</b> OTG_EN_A_VALID_FILTER</p>
Enable Filter on "b_valid" signal from PHY?	<p>Specifies whether to add a filter on the "b_valid" input from the PHY. If your PHY already has a filter on b_valid, then it is not necessary to enable the one in the DWC_otg. The filter is implemented in the DWC_otg_wpc module as an 8-bit counter that works on the PHY clock. In the case of the UTMI+ PHY, this pin is from the PHY. In the case of the ULPI PHY, this signal is generated by the ULPI Wrapper inside the core.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> (((OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1))) ? 1 : 0)</p> <p><b>Enabled:</b> ((OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Parameter Name:</b> OTG_EN_B_VALID_FILTER</p>

**Table 3-6 Additional Configuration Options Parameters (Continued)**

Label	Description
Enable Filter on "session_end" signal from PHY?	<p>Specifies whether to add a filter on the "session_end" input from the PHY. If your PHY already has a filter on session_end, then it is not necessary to enable the one in the DWC_otg. The filter is implemented in the DWC_otg_wpc module as an 8-bit counter that works on the PHY clock. In the case of the UTMI+ PHY, this pin is from the PHY. In the case of the ULPI PHY, this signal is generated by the ULPI Wrapper inside the core.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> (((OTG_MODE !=2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1))) ? 1 : 0)</p> <p><b>Enabled:</b> ((OTG_MODE !=2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Parameter Name:</b> OTG_EN_SESSIONEND_FILTER</p>
Enable Dynamic Power Reduction?	<p>Enable Active Clock Gating Support When enabled the controller internally gates the hclk, phy_clk and ram_clk to the majority of modules in the Controller. The clocks are gated during the Idle periods in between bursts of USB and AHB traffic This save dynamic power consumed by the controller</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ No (0)</li> <li>■ Yes (1)</li> </ul> <p><b>Default Value:</b> No</p> <p><b>Enabled:</b> OTG_MODE != 0 &amp;&amp; &lt;DWC-OTG-V4 feature authorized&gt;==1</p> <p><b>Parameter Name:</b> OTG_EN_ACG</p>
Enable UPF Power Switch Polarity?	<p>When Power Optimization features such as Partial Power Down, Hibernation and Extended Hibernation are enabled, this parameter is used to specify if Power Switch should of Logic High polarity or Logic Low polarity.</p> <ul style="list-style-type: none"> <li>■ When the parameter is set to '0', Power Switch will be in the instantiated with Logic Low polarity, that is, the switch will be OFF if the control signal is High, and the switch will be ON if the control signal is low.</li> <li>■ When the parameter is set to '1', Power Switch will be in the instantiated with Logic High polarity, that is, the switch will be ON if the control signal is High, and the switch will be OFF if the control signal is low.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ ActiveLow (0)</li> <li>■ ActiveHigh (1)</li> </ul> <p><b>Default Value:</b> ActiveLow</p> <p><b>Enabled:</b> OTG_EN_PWRLOPT!=0</p> <p><b>Parameter Name:</b> OTG_PWR_SWITCH_POLARITY</p>

## 3.7 Endpoint Direction Parameters

**Table 3-7 Endpoint Direction Parameters**

Label	Description
Direction of Endpoint n (for n = 1; n <= OTG_NUM_EPS)	<p>Specifies whether Endpoint n is IN and OUT, IN only, or OUT only. Control endpoints must be IN and OUT.</p> <p>If you specify an endpoint as IN and OUT, software can later use it either as a data or control endpoint. If you specify IN or OUT, later it can only be used as a data IN or OUT endpoint.</p> <p>An IN or OUT endpoint requires about 1K in register area.</p> <p>If area is not a concern and you want a flexible design that can be used in multiple applications, choose IN and OUT.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ IN and OUT (0)</li> <li>■ IN (1)</li> <li>■ OUT (2)</li> </ul> <p><b>Default Value:</b> IN and OUT</p> <p><b>Enabled:</b> OTG_MODE!=5 &amp;&amp; OTG_MODE!=6 &amp;&amp; OTG_NUM_EPS&gt;=1 &amp;&amp; ((OTG_NUM_CRL_EPS==0 &amp;&amp; OTG_EN_DESC_DMA==1)    (OTG_EN_DESC_DMA==0)) &amp;&amp; PHY_INST==0</p> <p><b>Parameter Name:</b> OTG_EP_DIR_n</p>

## 3.8 Device Periodic FIFO Depth Parameters

**Table 3-8 Device Periodic FIFO Depth Parameters**

Label	Description
Largest Device Mode Periodic Tx Data FIFO n Depth (4 to 768)  (for n = 1; n <= OTG_NUM_PERIO_EPS)	<p>Specifies the largest Device mode Periodic Tx Data FIFO n depth and the memory allocation for the device Periodic TxFIFO when dynamic FIFO sizing is disabled. This parameter also defines the device TxFIFO pointer size and should be set equal to the largest Device mode IN periodic packet. If dynamic FIFO sizing is enabled, software can set the device Periodic TxFIFO size up to the value specified for this parameter. For high-bandwidth endpoints, this should be set to the sum of all periodic packets to be sent in 1 microframe.</p> <p>For example, if you plan to send three 1 KB packets per microframe, use this value:  <math>3 * 1,024/4 = 768</math></p> <p><b>Note:</b> Device Periodic TxFIFO depths are changed by programming the respective FIFO start addresses.</p> <p>Please go through the Programming Guide for details on selecting the FIFO Size.</p> <p><b>Values:</b> 4, ..., 768</p> <p><b>Default Value:</b> 768</p> <p><b>Enabled:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 &amp;&amp; OTG_NUM_PERIO_EPS &gt;= 1</p> <p><b>Parameter Name:</b> OTG_TX_DPERIO_DFIFO_DEPTH_n</p>

## 3.9 Device IN Endpoint FIFO Depth Parameters

**Table 3-9 Device IN Endpoint FIFO Depth Parameters**

Label	Description
Largest Device Mode Tx Data FIFO n Depth (16 to 32768) (for n = 0; n <= OTG_NUM_IN_EPS-1)	<p>Specifies the largest Device mode Tx Data FIFO depth and the memory allocation for the device TxFIFO n. This parameter also defines the device TxFIFO n pointer size and should be set equal to the largest Device mode IN packet for the endpoint using this FIFO. If dynamic FIFO sizing is enabled, software can set the device TxFIFO n size up to the value specified for this parameter.</p> <p>The recommended depth is:  <math>(2 * \text{Maximum packet size in bytes} / 4 \text{ of the endpoint for which FIFO } \#n \text{ will be used})</math></p> <p>Although 1 maximum packet size (MPS) of memory is sufficient, allotting twice this amount allows simultaneous operation. That is, when the USB is reading one packet, the AHB can write the next packet. If the AHB latency is high or the AHB clock is slow, allot more than 2 MPS of memory to hold multiple transmit packet for the endpoint. If this FIFO is going to be re-programmed later to support different types endpoints (CTRL/BULK/ISOC), then choose the maximum of maximum packet size in the FIFO size calculation.</p> <p>Please go through the Programming Guide for details on selecting the FIFO Size.</p> <p><b>Values:</b> 16, ..., 32768</p> <p><b>Default Value:</b> 256</p> <p><b>Enabled:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 &amp;&amp; OTG_NUM_IN_EPS &gt;= 1 &amp;&amp; OTG_ENDED_TX_FIFO==1</p> <p><b>Parameter Name:</b> OTG_TX_DINEP_DFIFO_DEPTH_n</p>

### 3.10 Recommended Configuration Parameter Values

To gain more flexibility, you can use the recommended values for some of the parameters as listed in [Table 3-10](#). These recommendations are based on previous experience of usage. If you have any doubts in deciding a configuration parameter value for your target application, contact [Web Resources](#).



**Note** If your target application does not accommodate any of these recommendations, choose the parameter value that best suits your application, and not just the opposite value.

**Table 3-10 Recommended Configuration Parameter Values**

Configuration Parameter	Recommendation
OTG_MODE	Enable SRP in the following scenarios: <ul style="list-style-type: none"> <li>■ PHY provides the VBUS detection related signal (that is, utmisrp_bvalid)</li> <li>■ Self-powered Device mode is used</li> <li>■ Application does some special tasks when there is no connection (that is, advanced power strategy)</li> </ul>
OTG_ARCHITECTURE	Internal DMA (2)
OTG_ENDED_TX_FIFO	Yes (1)
OTG_EN_DESC_DMA	Yes (1)
OTG_HSPHY_INTERFACE	8/16 bits (2)
OTG_VENDOR_CTL_INTERFACE	Choose Yes(1) if ULPI PHY will be used and software wants to access PHY through ULPI interface also.
OTG_NUM_CRL_EPS	0
OTG_DFIFO_DYNAMIC	1
OTG_DFIFO_DEPTH	Review “Calculating FIFO Size” chapter in the Programming Guide. In general, for device, use <ul style="list-style-type: none"> <li>■ Method 2 for Bulk endpoints, and</li> <li>■ Method 3 with x=2 for high-bandwidth ISOC endpoints</li> </ul>
Other *_DEPTH parameters	If OTG_DFIFO_DYNAMIC is 1, set these parameters to the same value as OTG_DFIFO_DEPTH or a sufficiently large value because during initialization, software can only decrease the FIFO size from the configured value.
OTG_EN_*_FILTER	Enable filters if you are using Synopsys PHY which has no such filter.
OTG_EP_DIR_*	Use IN and OUT (0) if total gate count is acceptable.



# 4

## Signal Interfaces

---



**Note** For description of signals mentioned in this chapter, see [Chapter 5, “Signal Descriptions”](#).

---

This chapter describes the interface signals with which the DWC\_otg controller can be integrated into an application environment.

### 4.1 Signal Interfaces and Naming Conventions

There are several interfaces on the DWC\_otg controller, and each signal belonging to a particular interface has a unique prefix, listed here. Signals for these interfaces are described in the sections that follow.

s_h	AHB Slave interface
m_h	AHB Master interface
dma_	External DMA Controller interface
dfifo_	External Data FIFO (DFIFO) RAM interface
utmi_	UTMI+ Level 3 (L3) interface (except OTG USB 1.1 FS signals) and UTMI+ Vendor Control interface
utmiotg_	UTMI+ Level 3 (L3) OTG interface
utmisrp_	
utmifs_	UTMI+ USB 1.1 Full-Speed Serial Transceiver interface
ulpi_	ULPI interface
i2c_	I <sup>2</sup> C interface
ss_	Simulation Control interface
ic_usb	IC_USB interface
endp_	Endpoint-specific Interrupt
adp_	ADP related control signals
rid_	Battery charger related ACA signals
rms_	Remote memory support interface related signals



All signals with “\_n” are active low. For example, the utmi\_suspend\_n signal is an active low signal; that is, asserted low.

## 4.2 I/O Delays

Signal input/output delays are specified as a percent of their clocks, following the Synopsys Design Compiler format. These constraints are also available in coreConsultant.

AHB and PHY Resets	20% of clock
AHB Slave Interface	Inputs: 60–65% Outputs: 75%
AHB Master Interface	Inputs: 60% Outputs: 75%
External DMA Controller Interface	Inputs: 60% Outputs: 70–80%
External DMA FIFO RAM Interface	Inputs: 60% Outputs: 80%
UTMI+ Interface, UTMI+ Vendor Control Interface	Inputs: 50% Outputs: 75%; 40% for suspend signaling; 60% for data bus and control
ULPI Interface	Inputs: 60% Outputs: 40%; 60% for data output enable
I <sup>2</sup> C Interface	Inputs: 50% Outputs: 75%
IC_USB interface	Inputs: 50% Outputs: 60%
HSIC Interface (excluding UTMI+ signals)	Inputs: 40% Outputs: 40%
Simulation Control Interface	N/A (strap signals)
Miscellaneous Signals Interface	Inputs: 50% for GPIO, SOF toggle and counter; 20% for Scan mode port Outputs: 75% for GPIO, interrupt; 25% for internal probe (used in simulation environment)

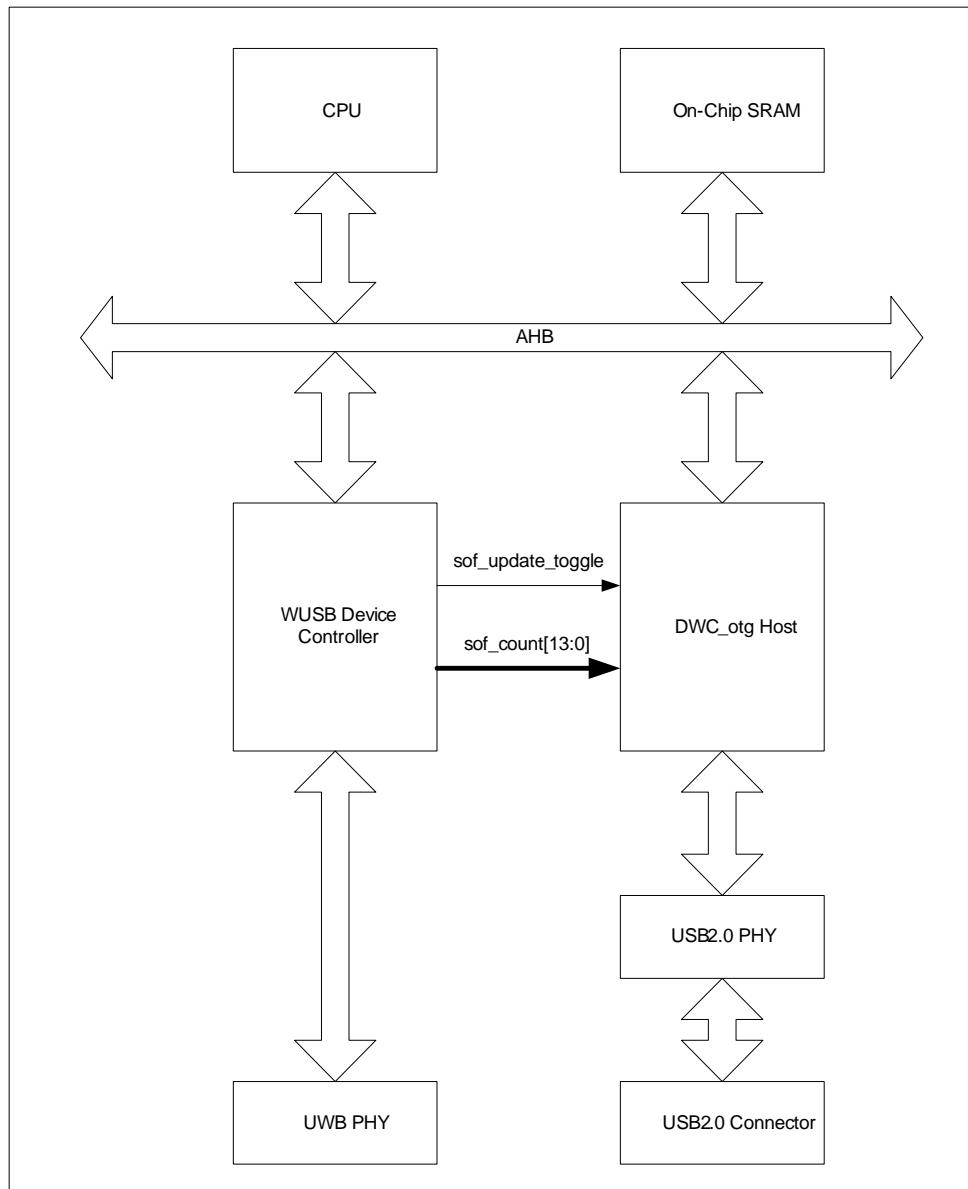
## 4.3 Device Wire Adaptor Signals

Device Wire Adaptor (DWA) is a short-term solution to convert a wired device to a wireless device. You can connect a wire adaptor to your existing wired printer to make it wireless. To build a DWA, you need a native wireless USB device controller and a wired USB host controller (such as DWC\_otg in host mode).

The SOF mechanism in DWC\_otg allows the transfer of wireless SOF time from a wireless-USB device to DWC\_otg host SOF timer. Input signals `sof_update_toggle` and `sof_count[13:0]` are used only in DWA applications. Other than this, DWC\_otg does not provide any wireless support.

[Figure 4-1](#) illustrates an example DWC\_otg DWA implementation.

**Figure 4-1 DWA Chip Architecture Example**



## 4.4 Debug Ports

The DWC\_otg controller has the following two top-level debug ports:

- internal\_probes, which is the debug port for signals in AHB clock domain
- internal\_probes\_p, which is the debug port for signals in PHY clock domain

The debug ports have information on the internal state of the controller, which can be used for debug purposes. The following sections describe the internal\_probes, and internal\_probes\_p debug ports, respectively:

- “[Debug Probes in AHB Clock Domain \(internal\\_probes\)](#)”
- “[Debug Probes in PHY Clock Domain \(internal\\_probes\\_p\)](#)” on page 174

### 4.4.1 Debug Probes in AHB Clock Domain (internal\_probes)

[Table 4-1](#) describes the mapping of the internal\_probes debug port.

**Table 4-1 Debug Probes in AHB Clock Domain**

Bit Position	Internal Signal	RTL Source File	Valid Modes
internal_probes[61:60]	hclk_state_cs	DWC_otg_acg.v	OTG_EN_ACG
internal_probes[59]	hclk_gate_en	DWC_otg_acg.v	OTG_EN_ACG
internal_probes[58]	dsch_chdis_push	DWC_otg_aiu_dsch.v	DESC_DMA_EN
internal_probes[57]	clr_last_pkt_xmit	DWC_otg_aiu_dsch.v	DESC_DMA_EN
internal_probes[56]	dsch_xmit_sel	DWC_otg_aiu_dsch.v	DESC_DMA_EN
internal_probes[55]	dsch_reqq_push	DWC_otg_aiu_dsch.v	DESC_DMA_EN
internal_probes[54:47]	dsch_tkn_req	DWC_otg_aiu_dsch.v	DESC_DMA_EN
internal_probes[46]	dsch_lp_done	DWC_otg_aiu_dsch.v	DESC_DMA_EN
internal_probes[45:44]	dsch_lp_req_type	DWC_otg_aiu_dsch.v	DESC_DMA_EN
internal_probes[43]	dsch_lp_req	DWC_otg_aiu_dsch.v	DESC_DMA_EN
internal_probes[42:28]	current_state	DWC_otg_aiu_dsch.v	DESC_DMA_EN
internal_probes[27]	tx_trans_start	DWC_otg_pfc_ssctl.v	All
internal_probes[26]	tx_trans_end	DWC_otg_pfc_ssctl.v	All
internal_probes[25]	txf_rd_rollover	DWC_otg_pfc_ssctl.v	All
internal_probes[24]	rx_sts_qual	DWC_otg_pfc_ssctl.v	All
internal_probes[23]	rx_trans_start	DWC_otg_pfc_ssctl.v	All
internal_probes[22:18]	rfx_numpkts[4:0]	DWC_otg_pfc_ssctl.v	All

**Table 4-1 Debug Probes in AHB Clock Domain (Continued)**

<b>Bit Position</b>	<b>Internal Signal</b>	<b>RTL Source File</b>	<b>Valid Modes</b>
internal_probes[17]	rx_data	DWC_otg_pfc_ssctl.v	All
internal_probes[16]	rx_nonpktsts	DWC_otg_pfc_ssctl.v	All
internal_probes[15:13]	rx_pktsts[2:0]	DWC_otg_pfc_ssctl.v	All
internal_probes[12]	rx_sts	DWC_otg_pfc_ssctl.v	All
internal_probes[11]	sp2hl_otg_status[1]	DWC_otg_core.v	All
internal_probes[10]	h2pd_16bit_phy_sel	DWC_otg_core.v	All
internal_probes[9]	h2pd_3pinfs_phy_sel	DWC_otg_core.v	All
internal_probes[8]	h2pd_fs_phy_sel	DWC_otg_core.v	All
internal_probes[7]	bius_rxsts_pop	DWC_otg_core.v	All
internal_probes[6]	bius_unpop_dfifo	DWC_otg_core.v	All
internal_probes[5]	bius_pop_dfifo	DWC_otg_core.v	All
internal_probes[4]	bius_push_dfifo	DWC_otg_core.v	All
internal_probes[3]	bius_read_datram	DWC_otg_core.v	All
internal_probes[2]	bius_write_datram	DWC_otg_core.v	All
internal_probes[1]	bius_read_csr	DWC_otg_core.v	All
internal_probes[0]	bius_write_csr	DWC_otg_core.v	All

#### 4.4.2 Debug Probes in PHY Clock Domain (internal\_probes\_p)

Table 4-2 describes the mapping of the internal\_probes\_p debug port.

**Table 4-2 Debug Probes in PHY Clock Domain**

Bit Position	Internal Signal	RTL Source File	Valid Modes
internal_probes_p[218:217]	phy_state_cs	DWC_otg_acg.v	OTG_EN_ACG
internal_probes_p[216]	phy_clk_gate_en	DWC_otg_acg.v	OTG_EN_ACG
internal_probes_p[215]	dbg_iso_tkn_end	DWC_otg_mac_pie.v	All
internal_probes_p[214:211]	curep[3:0]	DWC_otg_mac_pie.v	All
internal_probes_p[210]	txf_reqq_pop_dev	DWC_otg_pfc_tknresp.v	All
internal_probes_p[209]	src_pop	DWC_otg_pfc_tknresp.v	All
internal_probes_p[208]	sink_sts_push_dev	DWC_otg_pfc_tknresp.v	All
internal_probes_p[207]	sts_push	DWC_otg_pfc_tknresp.v	All
internal_probes_p[206]	short_pkt	DWC_otg_pfc_tknresp.v	All
internal_probes_p[205]	gout_nak_eff	DWC_otg_pfc_tknresp.v	All
internal_probes_p[204]	rewind_tobe_compl	DWC_otg_pfc_tknresp.v	All
internal_probes_p[203]	data_tobe_compl	DWC_otg_pfc_tknresp.v	All
internal_probes_p[202]	xfer_tobe_compl	DWC_otg_pfc_tknresp.v	All
internal_probes_p[201:198]	pkt_epnum	DWC_otg_pfc_tknresp.v	All
internal_probes_p[197:195]	pkt_sts	DWC_otg_pfc_tknresp.v	All
internal_probes_p[194]	txf_reqq_pop_hst	DWC_otg_pfc_tknreq.v	All
internal_probes_p[193]	sink_sts_push_hst	DWC_otg_pfc_tknreq.v	All
internal_probes_p[192:172]	p2hd_reg_status_hst	DWC_otg_pfc_tknreq.v	All
internal_probes_p[171]	ti_tkn_req	DWC_otg_pfc_tknreq.v	All
internal_probes_p[170:121]	ti_tkn_info	DWC_otg_pfc_tknreq.v	All
internal_probes_p[120:108]	current_state	DWC_otg_pfc_tknreq.v	All
internal_probes_p[107]	upi_rxvld	DWC_otg_mac_upi.v	All
internal_probes_p[106]	upi_rxvldh	DWC_otg_mac_upi.v	All

**Table 4-2 Debug Probes in PHY Clock Domain**

internal_probes_p[105]	upi_rxactive	DWC_otg_mac_upi.v	All
internal_probes_p[104]	upi_rxerr	DWC_otg_mac_upi.v	All
internal_probes_p[103]	upi_txrdy	DWC_otg_mac_upi.v	All
internal_probes_p[102]	upi_bus_idle	DWC_otg_mac_upi.v	All
internal_probes_p[101]	upi_line_se0	DWC_otg_mac_upi.v	All
internal_probes_p[100:99]	prt_clk_sel	DWC_otg_mac_upi.v	All
internal_probes_p[98:97]	dev_clk_sel	DWC_otg_mac_upi.v	All
internal_probes_p[96:95]	rx_state	DWC_otg_mac_upi.v	HSPHY_DWIDTH_16_O NLY=0
internal_probes_p[94:92]	tx_state	DWC_otg_mac_upi.v	HSPHY_DWIDTH_16_O NLY=0
internal_probes_p[91]	if_dev_mode	DWC_otg_mac_otgif.v	INSTANTIATE_OTG
internal_probes_p[90]	if_overcurr_det	DWC_otg_mac_otgif.v	INSTANTIATE_OTG
internal_probes_p[89]	if_srp_req_det	DWC_otg_mac_otgif.v	INSTANTIATE_OTG
internal_probes_p[88]	p2ht_otg_sts_rdy	DWC_otg_mac_otgif.v	INSTANTIATE_OTG
internal_probes_p[87:85]	p2ht_otg_status	DWC_otg_mac_otgif.v	INSTANTIATE_OTG
internal_probes_p[84]	if_dline_pulsing	DWC_otg_mac_otgif.v	INSTANTIATE_OTG
internal_probes_p[83]	if_conn_ena	DWC_otg_mac_otgif.v	INSTANTIATE_OTG
internal_probes_p[82]	if_peri_state	DWC_otg_mac_otgif.v	INSTANTIATE_OTG
internal_probes_p[81]	if_hst_state	DWC_otg_mac_otgif.v	INSTANTIATE_OTG
internal_probes_p[80:79]	p2hd_otg_status	DWC_otg_mac_otgif.v	INSTANTIATE_OTG
internal_probes_p[78:75]	d_state	DWC_otg_mac_otgif.v	INSTANTIATE_OTG
internal_probes_p[74]	Unused	DWC_otg_mac_dssr.v	INSTANTIATE_DEVICE
internal_probes_p[73]	dev_in_resume	DWC_otg_mac_dssr.v	INSTANTIATE_DEVICE
internal_probes_p[72]	dev_in_suspend	DWC_otg_mac_dssr.v	INSTANTIATE_DEVICE
internal_probes_p[71]	dev_in_wakeup	DWC_otg_mac_dssr.v	INSTANTIATE_DEVICE

**Table 4-2 Debug Probes in PHY Clock Domain**

internal_probes_p[70]	dev_in_reset	DWC_otg_mac_dssr.v	INSTANTIATE_DEVICE
internal_probes_p[69]	dev_in_idle	DWC_otg_mac_dssr.v	INSTANTIATE_DEVICE
internal_probes_p[68]	dev_resume_det	DWC_otg_mac_dssr.v	INSTANTIATE_DEVICE
internal_probes_p[67]	sr_suspend_3ms	DWC_otg_mac_dssr.v	INSTANTIATE_DEVICE
internal_probes_p[66]	hnp_success	DWC_otg_mac_dssr.v	INSTANTIATE_DEVICE
internal_probes_p[65:64]	enum_spd	DWC_otg_mac_dssr.v	INSTANTIATE_DEVICE
internal_probes_p[63:60]	dssr_state	DWC_otg_mac_dssr.v	INSTANTIATE_DEVICE
internal_probes_p[59]	dbg_hppt_babble	DWC_otg_mac_prt.v	INSTANTIATE_HOST
internal_probes_p[58]	dbg_hppt_con_sts	DWC_otg_mac_prt.v	INSTANTIATE_HOST
internal_probes_p[57]	dbg_hppt_soft_dis	DWC_otg_mac_prt.v	INSTANTIATE_HOST
internal_probes_p[56]	dbg_hppt_in_reset	DWC_otg_mac_prt.v	INSTANTIATE_HOST
internal_probes_p[55]	dbg_hppt_in_resume	DWC_otg_mac_prt.v	INSTANTIATE_HOST
internal_probes_p[54]	dbg_hppt_suspend	DWC_otg_mac_prt.v	INSTANTIATE_HOST
internal_probes_p[53]	dbg_hppt_enable	DWC_otg_mac_prt.v	INSTANTIATE_HOST
internal_probes_p[52:51]	dbg_hppt_enum_spd	DWC_otg_mac_prt.v	INSTANTIATE_HOST
internal_probes_p[50]	dbg_hppt_rmwkup_det	DWC_otg_mac_prt.v	INSTANTIATE_HOST
internal_probes_p[49:45]	prt_state	DWC_otg_mac_prt.v	INSTANTIATE_HOST
internal_probes_p[44]	upi_rxvld	DWC_otg_mac.v	All
internal_probes_p[43]	fifo_send_nak   early_tkn	DWC_otg_mac_pie.v	DESC_DMA_EN
internal_probes_p[42]	(txrx_state==WAIT_DATA) & (ti_txdata[35:32]==4'd0) & ti_txdata_avail)	DWC_otg_mac_pie.v	DESC_DMA_EN
internal_probes_p[42]	ti_inep_zerolen	DWC_otg_mac_pie.v	!DESC_DMA_EN

**Table 4-2 Debug Probes in PHY Clock Domain**

internal_probes_p[41]	send_null_pkt	DWC_otg_mac_pie.v	DESC_DMA_EN
internal_probes_p[40]	rx_over_flow	DWC_otg_mac_pie.v	DESC_DMA_EN
internal_probes_p[39]	dbg_isoin_nodata	DWC_otg_mac_pie.v	All
internal_probes_p[38]	dbg_csplit_hshk	DWC_otg_mac_pie.v	All
internal_probes_p[37]	dbg_protocol_stall	DWC_otg_mac_pie.v	All
internal_probes_p[36]	dbg_nak_setup	DWC_otg_mac_pie.v	All
internal_probes_p[35]	dbg_no_resp	DWC_otg_mac_pie.v	All
internal_probes_p[34]	p2ht_isopkt_drop	DWC_otg_core.v	All
internal_probes_p[33]	mac_rxvalid	DWC_otg_core.v	All
internal_probes_p[32]	dbg_hdsk_pidchk	DWC_otg_mac_pie.v	DESC_DMA_EN
internal_probes_p[31]	p2hl_tx_undrun	DWC_otg_core.v	DESC_DMA_EN
internal_probes_p[30]	dbg_lostack_retry	DWC_otg_mac_pie.v	DESC_DMA_EN
internal_probes_p[29]	dbg_hdsk_rxerr	DWC_otg_mac_pie.v	DESC_DMA_EN
internal_probes_p[28]	dbg_pkt_babble	DWC_otg_mac_pie.v	DESC_DMA_EN
internal_probes_p[27]	dbg_data_crc	DWC_otg_mac_pie.v	DESC_DMA_EN
internal_probes_p[26:19]	wpc_dataout	DWC_otg_core.v	All
internal_probes_p[18]	wpc_drvvbus	DWC_otg_core.v	All
internal_probes_p[17]	dbg_data_pidchk	DWC_otg_mac_pie.v	All
internal_probes_p[16]	dbg_data_rxerr	DWC_otg_mac_pie.v	All
internal_probes_p[15]	dbg_inter_pkt	DWC_otg_mac_pie.v	All
internal_probes_p[14]	dbg_dev_exist	DWC_otg_mac_pie.v	All
internal_probes_p[13]	dbg_ep_exist	DWC_otg_mac_pie.v	All
internal_probes_p[12]	dbg_tkn_crc	DWC_otg_mac_pie.v	All
internal_probes_p[11]	dbg_tkn_pidchk	DWC_otg_mac_pie.v	All
internal_probes_p[10]	dbg_tkn_rxerr	DWC_otg_mac_pie.v	All

**Table 4-2 Debug Probes in PHY Clock Domain**

internal_probes_p[9:8]	tkn_state	DWC_otg_mac_pie.v	All
internal_probes_p[7:5]	rcv_state	DWC_otg_mac_pie.v	All
internal_probes_p[4:0]	txrx_state	DWC_otg_mac_pie.v	All

## 4.5 Overriding Control of PHY Voltage Valid and ID Input Signals

You can choose to override the values of the following signals and enable the application to control them:

- utmiotg\_iddig
- utmiotg\_avalid
- utmisrp\_bvalid
- utmiotg\_vbusvalid

The following register bits are required for each of the above input signals:

- **override\_enable** bit: This bit can be used to enable or disable the software over-ride functionality enabling the DWC\_otg controller to be controlled by an application or by the PHY inputs. When set to 1, the application controls the signal. When set to 0, the signal is controlled by the PHY input.
- **override\_values** bit: This bit can be used to control the value of the application override when the signal is controlled by the application.
- **utmiotg\_iddig** bit:
  - Force Device Mode: Writing a 1 to this bit overrides the value of utmiotg\_iddig input pin with value 1.
  - Force Host Mode: Writing a 1 to this bit overrides the value of utmiotg\_iddig input pin with value 0.



**Note** The **override\_enable** and **override\_values** are generically named. Each signal has a specific register bit that controls it.

### Limitations of this Feature

- You must first program **override\_values** bit to the desired value. Wait for 5 PHY clocks and then program **override\_enable** to 1 to enable the override. After this initial programming, the **override\_value** can be changed to any value multiple times.
- Because synchronization to PHY clock is required for these register bits, the override functionality is not available when PHY clock is off (for example, during Suspend or Session-off states when **phy\_clk** is off).
- The override functionality is not available if the controller is in hibernation. The override values programmed before hibernation are retained during hibernation. The application must first switch on the controller and then use this function.



## 5

# Signal Descriptions

This chapter details all possible I/O signals in the controller. For configurable IP titles, your actual configuration might not contain all of these signals.

Inputs are on the left of the signal diagrams; outputs are on the right.

**Attention: For configurable IP titles, do not use this document to determine the exact I/O footprint of the controller. It is for reference purposes only.**

When you configure the controller in coreConsultant, you must access the I/O signals for your actual configuration at workspace/report/IO.html or workspace/report/IO.xml after you have completed the report creation activity. That report comes from the exact same source as this chapter but removes all the I/O signals that are not in your actual configuration. This does not apply to non-configurable IP titles. In addition, all parameter expressions are evaluated to actual values. Therefore, the widths might change depending on your actual configuration.

Some expressions might refer to TCL functions or procedures (sometimes identified as <functionof>) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the controller in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

In addition to describing the function of each signal, the signal descriptions in this chapter include the following information:

**Active State:** Indicates whether the signal is active high or active low. When a signal is not intended to be used in a particular application, then this signal needs to be tied or driven to the inactive state (opposite of the active state).

**Registered:** Indicates whether or not the signal is registered directly inside the IP boundary without intervening logic (excluding simple buffers). A value of No does not imply that the signal is not synchronous, only that there is some combinatorial logic between the signal's origin or destination register and the boundary of the controller. A value of N/A indicates that this information is not provided for this IP title.

**Synchronous to:** Indicates which clocks in the IP sample this input (drive for an output) when considering all possible configurations. A particular configuration might not have all of the clocks listed. This clock might not be the same as the clock that your application logic should use to clock (sample/drive) this pin. For more details, consult the clock section in the databook.

**Exists:** Names of configuration parameters that populate this signal in your configuration.

**Validated by:** Assertion or de-assertion of signals that validates the signal being described.

### Additional Details on Synchronous To:

Synchronous to attribute indicates which controller clock(s) samples an input (or drives an output). It is a parameterized expression which evaluates to a clock(s) that samples input (or drives an output) in a configuration.

- In many cases, an output has components driven (or an input sampled) by multiple clocks.
- Upon evaluation, when there is more than one clock in the list:
  - A {} is placed around the clock which (by default first clock in list) is used in generating the synthesis in/out timing delay constraints and is also used in Spyglass boundary CDC checking.
  - Though the evaluated clock list might contain multiple clocks, only a single clock would be active as configured by the programmable bits, or they are internally synchronous to each other.
  - In most cases, this is also the clock that you should use to drive/sample the pin in normal operational (non low-power) mode.

Example Synchronous to expression showing clock used for Synthesis in/out timing delay constraints and Spyglass CDC checking:

```
((OTG_EN_PWR0PT == 2) || (OTG_EN_PWR0PT == 3) || (OTG_AD_P SUPPORT == 1)) ?  
"pmu_hclk,{hclk}":"hclk"
```

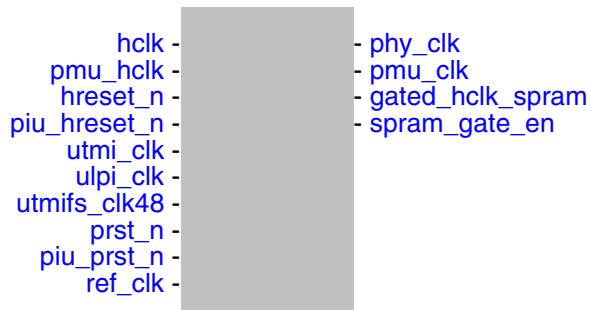
- When you see per-bit clock format such as "0:0=None;2:2=hclk\_gated;3:3=None" in the clock list, it indicates that some of the bits in the multi-bit port have different clocks, and/or some of the bits fall into various categories as defined next by "None".
- "None" in the evaluated clock list indicates any of the following and is equivalent/mapped to a virtual\_clk for CDC and synthesis.
  - Direct or combinatorial feed -through
  - Gated-off (that is, unclocked) inputs
  - Unused inputs
  - Hard-coded outputs
  - Asynchronous outputs
  - Asynchronous resets

The I/O signals are grouped as follows:

- Common Clock and Reset Signals on [page 184](#)
- AHB Slave Interface Signals on [page 189](#)
- AHB Master Interface Signals on [page 194](#)
- External DMA Controller Interface Signals on [page 199](#)
- External Data FIFO (DFIFO) RAM Interface Signals on [page 202](#)
- UTMI+ Parallel Interface Signals on [page 206](#)
- UTMI+ PMU Interface Signals on [page 228](#)

- UTMI+ OTG Interface Signals on [page 232](#)
- UTMI+ Vendor Control Interface Signals on [page 242](#)
- ULPI Interface Signals on [page 244](#)
- I2C Interface Signals on [page 249](#)
- LPM Interface Signals on [page 251](#)
- HSIC Interface Signals on [page 254](#)
- IC\_USB Interface Signals on [page 255](#)
- Simulation Control Interface Signals on [page 267](#)
- Miscellaneous Interface Signals on [page 268](#)
- Remote Memory Support Interface Signals on [page 275](#)
- ADP Interface Signals on [page 278](#)
- Battery Charger ACA Interface Signals on [page 280](#)

## 5.1 Common Clock and Reset Signals



**Table 5-1 Common Clock and Reset Signals**

Port Name	I/O	Description
hclk	I	<p>AHB Clock. Clock from application bus. This signal must be greater than or equal to 30 MHz.</p> <p><b>Note:</b> The AHB Frequency and PHY Interface Data Width determine the number of levels of cascaded Hubs that can be supported by the Controller. Please refer to the section "Impact of System Clocks on Minimum Inter-Packet Gap and Number of Cascaded Hubs".</p> <p><b>See Also:</b> To gate this clock for power optimization, see 'FIFO RAM Allocation' in the Programming Guide.</p> <p><b>Exists:</b> Always <b>Synchronous To:</b> None <b>Registered:</b> N/A <b>Power Domain:</b> {OTG_EN_PWRROPT==1 ? "Vdd" : OTG_EN_PWRROPT==2 ? "Vsw" : OTG_EN_PWRROPT==3 ? "Vdd" : "Vdd"} <b>Active State:</b> N/A</p>
pmu_hclk	I	<p>PMU AHB Clock. Clock to PMU module when hibernation and/or ADP is enabled. Must be greater than or equal to 15 MHz.</p> <p><b>Exists:</b> ((OTG_EN_PWRROPT == 2)    (OTG_EN_PWRROPT == 3)    (OTG_ADH_SUPPORT == 1))</p> <p><b>Synchronous To:</b> None <b>Registered:</b> N/A <b>Power Domain:</b> Vdd <b>Active State:</b> N/A</p>

**Table 5-1 Common Clock and Reset Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
hreset_n	I	AHB Clock Domain Reset. System reset synchronized to hclk. <b>Exists:</b> Always <b>Synchronous To:</b> None <b>Registered:</b> OTG_SYNC_RESET_TYPE==1 ?"No":"N/A" <b>Power Domain:</b> Vdd <b>Active State:</b> Low
piu_hreset_n	I	AHB Clock Domain Reset. System reset synchronized to hclk when OTG_EN_PWRROPT = 3. <b>Exists:</b> ((OTG_EN_PWRROPT == 3) && OTG_MODE !=5 && OTG_MODE !=6) <b>Synchronous To:</b> None <b>Registered:</b> N/A <b>Power Domain:</b> Vdd <b>Active State:</b> Low
utmi_clk	I	UTMI Clock. Receives the 30 MHz or 60 MHz clock supplied by the High Speed UTMI+ PHY. This clock input should be used as the Test clock. <b>Exists:</b> Always <b>Synchronous To:</b> None <b>Registered:</b> N/A <b>Power Domain:</b> Vdd <b>Active State:</b> N/A
ulpi_clk	I	ULPI Clock. Receives the 60-MHz clock supplied by the High Speed ULPI PHY. The negative edge is also used in DDR mode. <b>Exists:</b> (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3) <b>Synchronous To:</b> None <b>Registered:</b> N/A <b>Power Domain:</b> Vdd <b>Active State:</b> N/A
utmifs_clk48	I	FS Shared-Pin Interface. Used as utmifs_clk48, the USB 1.1 FS clock. See 'UTMI+ PMU Interface' for details. <b>Exists:</b> (OTG_FSPHY_INTERFACE != 0) <b>Synchronous To:</b> None <b>Registered:</b> N/A <b>Power Domain:</b> Vdd <b>Active State:</b> N/A

**Table 5-1 Common Clock and Reset Signals (Continued)**

Port Name	I/O	Description
prst_n	I	<p>PHY Clock Domain Reset. System reset synchronized to the following:</p> <ul style="list-style-type: none"> <li>■ utmi_clk</li> <li>■ ulpi_clk</li> <li>■ utmifs_clk48</li> </ul> <p><b>Exists:</b> Always  <b>Synchronous To:</b> None  <b>Registered:</b> OTG_SYNC_RESET_TYPE==1 ?"Yes":"N/A"  <b>Power Domain:</b> Vdd  <b>Active State:</b> Low</p>
piu_prst_n	I	<p>PHY Clock Domain Reset. System reset synchronized to:</p> <ul style="list-style-type: none"> <li>■ utmi_clk</li> <li>■ ulpi_clk</li> <li>■ utmifs_clk48</li> </ul> <p><b>Exists:</b> ((OTG_EN_PWROPT == 3) &amp;&amp; OTG_MODE !=5 &amp;&amp; OTG_MODE !=6)  <b>Synchronous To:</b> None  <b>Registered:</b> N/A  <b>Power Domain:</b> Vdd  <b>Active State:</b> Low</p>
phy_clk	O	<p>PHY Clock Output. Generated clock based on one of the following inputs:</p> <ul style="list-style-type: none"> <li>■ utmi_clk</li> <li>■ ulpi_clk</li> <li>■ utmifs_clk48</li> </ul> <p><b>Exists:</b> Always  <b>Synchronous To:</b> None  <b>Registered:</b> No  <b>Power Domain:</b> Vdd  <b>Active State:</b> N/A</p>

**Table 5-1 Common Clock and Reset Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
pmu_clk	O	<p>PMU PHY Clock Output. Generated clock based on one of the following inputs and also based which PHY interface is selected.</p> <ul style="list-style-type: none"> <li>■ utmi_clk</li> <li>■ ulpi_clk</li> <li>■ utmifs_clk48</li> </ul> <p>The clock is valid only when the Hibernation feature is enabled. Otherwise, it reads 1'b0. The clock can be used for debug purposes in Hibernation.</p> <p><b>Exists:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT == 1))</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> Not applicable</p>
ref_clk	I	<p>Reference clock for ADC 3.0 support</p> <p>This clock is used to run the frame/microframe counters when the GREFCLK.RefClkMode bit is set to 1'b1.</p> <p>The frequencies supported for this clock are 12, 16, 17, 19.2, 20, 24, 30, and 40 MHz.</p> <p><b>Exists:</b> (OTG_SERV_INT_ENH == 1)</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> Not applicable</p>
gated_hclk_spram	O	<p>RAM Clock which is a gated version of the AHB Clock</p> <p>This signal is available only if the Active Clock Gating Feature is enabled.</p> <p><b>Exists:</b> OTG_EN_ACG==1</p> <p><b>Synchronous To:</b> hclk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vsw" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> N/A</p>

**Table 5-1 Common Clock and Reset Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
spram_gate_en	O	<p>RAM Clock Gating Enable signal When asserted, it indicates that the RAM Clock output "gated_hclk_sram" is gated This signal is available only if the Active Clock Gating Feature is enabled.</p> <p><b>Exists:</b> OTG_EN_ACG==1 <b>Synchronous To:</b> hclk,hclk_gated <b>Registered:</b> No <b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vsw" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"} <b>Active State:</b> N/A</p>

## 5.2 AHB Slave Interface Signals

s_haddr	-	s_hready_resp
s_hsel	-	s_hresp
s_hwwrite	-	s_hrdtata
s_htrans	-	
s_hsize	-	
s_hburst	-	
s_hready	-	
s_hwdata	-	
s_hbig endian	-	

This interface connects the DWC\_otg core as a Slave on the AHB and is used by the microcontroller (AHB Master) to read from and write to core CSRs. Features supported on this interface include:

- Supports INCR4, INCR8, INCR16, INCR and SINGLE transfers
- 32-bit transfers
- OKAY response

**Table 5-2 AHB Slave Interface Signals**

Port Name	I/O	Description
s_hready_resp	O	<p>AHB Ready Response. Asserted by the DWC_otg core completion of a data transfer.</p> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? "pmu_hclk,{hclk}":"hclk"</p> <p><b>Registered:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? "No": "Yes"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-2 AHB Slave Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
s_hresp[1:0]	O	<p>AHB Response Type. Indicates transfer response.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 2'b00: OKAY</li> <li>■ Other values reserved.</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> hclk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
s_hrdata[31:0]	O	<p>AHB Read Data. Application read data bus</p> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? "pmu_hclk,{hclk}":"hclk"</p> <p><b>Registered:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? "No": "Yes"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
s_haddr[31:0]	I	<p>AHB Address. AHB address bus</p> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? "0:1=hclk;2:31=pmu_hclk,{hclk}":"hclk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-2 AHB Slave Interface Signals (Continued)**

Port Name	I/O	Description
s_hsel	I	<p>AHB Slave Select. Asserted when the transaction address falls within the memory address range allocated in the DWC_otg core.</p> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? "pmu_hclk,{hclk}" : "hclk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
s_hwwrite	I	<p>AHB Read/Write Command. Indicates whether the current transaction is a read or write.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Read</li> <li>■ 1'b1: Write</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? "pmu_hclk,{hclk}" : "hclk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
s_htrans[1:0]	I	<p>AHB Transfer Type. Indicates AHB transfer type.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 2'b00: IDLE</li> <li>■ 2'b01: BUSY</li> <li>■ 2'b10: NONSEQ</li> <li>■ 2'b11: SEQ</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? "pmu_hclk,{hclk}" : "hclk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-2 AHB Slave Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
s_hsize[2:0]	I	<p>AHB Data Transfer Size. Indicates AHB transfer size.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 3'b010: 32 bits</li> <li>■ Other values reserved.</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> hclk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
s_hburst[2:0]	I	<p>AHB Burst Size. Indicates the AHB burst size.</p> <p><b>Note:</b> This signal is input from the AHB fabric, the DWC_otg slave does not use this input to decode the burst type for the transaction. It is configured by the application.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 3'b000: SINGLE</li> <li>■ 3'b001: INCR</li> <li>■ 3'b011: INCR4</li> <li>■ 3'b101: INCR8</li> <li>■ 3'b111: INCR16</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
s_hready	I	<p>AHB Ready In. The HREADY signal from the currently selected AHB Slave. It is asserted to indicate a completed data transfer.</p> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P SUPPORT == 1)) ? "pmu_hclk,{hclk}" : "hclk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-2 AHB Slave Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
s_hwdata[31:0]	I	<p>AHB Write Data. Application write data bus</p> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_ADP_SUPPORT == 1)) ? "pmu_hclk,hclk_gated,{hclk}":"hclk,hclk_gated"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
s_hbig endian	I	<p>AHB Slave Big Endian Mode. Indicates the AHB Endian mode.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Little</li> <li>■ 1'b1: Big</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_ADP_SUPPORT == 1)) ? "pmu_hclk,{hclk}":"hclk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

## 5.3 AHB Master Interface Signals



This interface connects the DWC\_otg core as a Master on the AHB and is used by the built-in DMA Engine to access the system memory on the AHB. This interface:

- Does not generate locked transfers
- Does not generate wrapped burst transfers

**Table 5-3 AHB Master Interface Signals**

Port Name	I/O	Description
m_hbusreq	O	<p>AHB Request. Asserted by the core to request AHB grant for memory access.</p> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "hclk,hclk_gated" : "hclk_gated") : "hclk_gated"</p> <p><b>Registered:</b> {OTG_PWR_CLAMP} == 1 ? "Yes" : "No"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-3 AHB Master Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
m_hwwrite	O	<p>AHB Read/Write Command. Indicates whether the current transaction is a read or write request.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Read</li> <li>■ 1'b1: Write</li> </ul> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "hclk,hclk_gated" : "hclk_gated") : "hclk_gated"</p> <p><b>Registered:</b> {OTG_PWR_CLAMP} == 1 ? "Yes" : "No"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
m_haddr[31:0]	O	<p>AHB Address. Provides the address the core reads or writes.</p> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "hclk,hclk_gated" : "hclk_gated") : "hclk_gated"</p> <p><b>Registered:</b> {OTG_PWR_CLAMP} == 1 ? "Yes" : "No"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
m_htrans[1:0]	O	<p>AHB Transfer Type. Indicates the data transfer type.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 2'b00: IDLE</li> <li>■ 2'b01: BUSY</li> <li>■ 2'b10: NONSEQ</li> <li>■ 2'b11: SEQ</li> </ul> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "hclk,hclk_gated" : "hclk_gated") : "hclk_gated"</p> <p><b>Registered:</b> {OTG_PWR_CLAMP} == 1 ? "Yes" : "No"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-3 AHB Master Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
m_hsize[2:0]	O	<p>AHB Data Transfer Size. Indicates the data transfer size.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 3'b000: 8 bits</li> <li>■ 3'b010: 32 bits</li> <li>■ Other values reserved.</li> </ul> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "0:0=None;2:2=hclk_gated;3:3=None" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "0:0=None;2:2=hclk,hclk_gated;3:3=None" : "0:0=None;2:2=hclk_gated;3:3=None") : "0:0=None;2:2=hclk_gated;3:3=None"</p> <p><b>Registered:</b> {OTG_PWR_CLAMP} == 1 ? "Yes" :"No"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
m_hburst[2:0]	O	<p>AHB Burst Type. Indicates the bursting mode used by the AHB Master during a burst transaction.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 3'b000: SINGLE</li> <li>■ 3'b001: INCR</li> <li>■ 3'b011: INCR4</li> <li>■ 3'b101: INCR8</li> <li>■ 3'b111: INCR16 No other values are supported.</li> </ul> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "hclk,hclk_gated" : "hclk_gated") : "hclk_gated"</p> <p><b>Registered:</b> {OTG_PWR_CLAMP} == 1 ? "Yes" :"No"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-3 AHB Master Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
m_hprot[3:0]	O	<p>AHB Protection Control. Provides information bus access that is implemented with some level of protection.</p> <p><b>Values:</b> This signal is always tied to 4'b0001, indicating data access.</p> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "None" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "hclk" : "None") : "None"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
m_hwdata[31:0]	O	<p>AHB Write Data. Data bus for core writes to memory. Valid data is present only the byte lanes represented by the lower two bits of m_haddr when the transfer size is less than 32 bits.</p> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "hclk,hclk_gated" : "hclk_gated") : "hclk_gated"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
m_hgrant	I	<p>AHB Grant. Asserted in response to m_hbusreq, indicating that the DWC_otg core has gained ownership of the AHB.</p> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> hclk_gated</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==0 ? "Vdd" : "Vsw"}</p> <p><b>Active State:</b> High</p>
m_hready	I	<p>AHB Ready. Asserted by the AHB Slave (system memory) to indicate completion of a data transfer.</p> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> hclk_gated</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==0 ? "Vdd" : "Vsw"}</p> <p><b>Active State:</b> High</p>

**Table 5-3 AHB Master Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
m_hodata[31:0]	I	<p>AHB Read Data. Application read data bus.</p> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> hclk_gated</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==0 ? "Vdd" : "Vsw"}</p> <p><b>Active State:</b> High</p>
m_hresp[1:0]	I	<p>AHB Transfer Response. Indicates the response of the AHB Slave to the data transfer. This signal is qualified by m_hready.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 2'b00: OKAY</li> <li>■ 2'b01: ERROR</li> <li>■ 2'b10: RETRY</li> <li>■ 2'b11: SPLIT</li> </ul> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> hclk_gated</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==0 ? "Vdd" : "Vsw"}</p> <p><b>Active State:</b> High</p>
m_hbig endian	I	<p>AHB Master Big Endian Mode. Indicates the AHB Endian mode.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Little</li> <li>■ 1'b1: Big</li> </ul> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> hclk_gated</p> <p><b>Registered:</b> (OTG_SYNC_RESET_TYPE ==1 )? "No":"Yes"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==0 ? "Vdd" : "Vsw"}</p> <p><b>Active State:</b> High</p>

## 5.4 External DMA Controller Interface Signals



This interface connects the DWC\_otg core to an external DMA controller. Depending on the controller selected, an additional external wrapper can be required.

**Table 5-4 External DMA Controller Interface Signals**

Port Name	I/O	Description
dma_req	O	<p>DMA Request. Indicates an external DMA transfer between the TxFIFO/RxFIFO and system memory.</p> <ul style="list-style-type: none"> <li>■ When asserted, this signal qualifies dma_burst, dma_channel, and dma_count.</li> <li>■ It is de-asserted the clock following the dma_done strobe.</li> </ul> <p><b>Exists:</b> OTG_ARCHITECTURE == 1  <b>Synchronous To:</b> (OTG_EN_PWROPT==0    OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : (OTG_PWR_CLAMP==1 ? "hclk_gated" : "{hclk},hclk_gated")  <b>Registered:</b> No  <b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}  <b>Active State:</b> High</p>

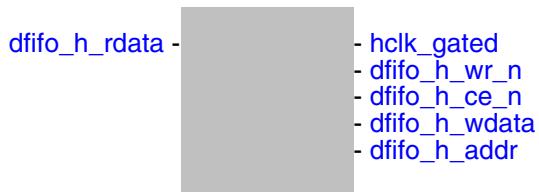
**Table 5-4 External DMA Controller Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
dma_channel[4:0]	O	<p>DMA Channel/Endpoint Number. Provides the number of the host channel or device endpoint requesting DMA. This field is qualified with dma_req asserted.</p> <p><b>Host Mode:</b></p> <ul style="list-style-type: none"> <li>■ Bits[3:0]: Active host channel requesting DMA</li> <li>■ Bit[4]: <ul style="list-style-type: none"> <li>- 1'b0: IN channel (write to AHB)</li> <li>- 1'b1: OUT Channel (read from AHB)</li> </ul> </li> <li>■ Bit [4] always determines IN or OUT no matter how many channels you have defined in bit [3].</li> <li>■ A channel can be only direction IN or OUT at a given time. This reflects the value in register field HCCHARn.EPDir.</li> </ul> <p><b>Device mode:</b></p> <ul style="list-style-type: none"> <li>■ Bits[3:0]: Active Endpoint Requesting DMA</li> <li>■ Bit[4]: <ul style="list-style-type: none"> <li>- 1'b0: OUT endpoint (write to AHB)</li> <li>- 1'b1: IN endpoint (read from AHB)</li> </ul> </li> <li>■ Specify endpoint direction using parameter OTG_EP_DIR_n. Synopsys recommends that you configure OTG_EP_DIR_n = 0 (bi-directional endpoint)</li> </ul> <p><b>Exists:</b> OTG_ARCHITECTURE == 1  <b>Synchronous To:</b> (OTG_EN_PWROPT==0    OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : (OTG_PWR_CLAMP==1 ? "hclk_gated" : {"hclk",hclk_gated"})  <b>Registered:</b> ((OTG_EN_PWROPT != 0) &amp;&amp; (OTG_PWR_CLAMP == 1)) ? "Yes" : "No"  <b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}  <b>Active State:</b> High</p>
dma_count[10:0]	O	<p>DMA Request Count. Indicates the DMA data transfer size in bytes. This signal is qualified with dma_req asserted.</p> <p><b>Exists:</b> OTG_ARCHITECTURE == 1  <b>Synchronous To:</b> hclk_gated  <b>Registered:</b> ((OTG_EN_PWROPT != 0) &amp;&amp; (OTG_PWR_CLAMP == 1)) ? "Yes" : "No"  <b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}  <b>Active State:</b> High</p>

**Table 5-4 External DMA Controller Interface Signals (Continued)**

Port Name	I/O	Description
dma_burst[3:0]	O	<p>DMA Burst Size. This reflects the value in register field GAHBCFG.HBstLen (Burst Length).</p> <p>Typically, an external wrapper uses this signal to interface to a standard DMA controller like Synopsys DW_ahb_dmac or ARM PrimeCell.</p> <p>This signal is qualified with dma_req asserted.</p> <p><b>Exists:</b> OTG_ARCHITECTURE == 1</p> <p><b>Synchronous To:</b> hclk_gated</p> <p><b>Registered:</b> ((OTG_EN_PWROPT != 0) &amp;&amp; (OTG_PWR_CLAMP == 1)) ? "Yes" : "No"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
dma_done	I	<p>DMA Done. Indicates completion of a DMA data transfer through the AHB Slave interface.</p> <p><b>Exists:</b> OTG_ARCHITECTURE == 1</p> <p><b>Synchronous To:</b> hclk_gated</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==0 ? "Vdd" : "Vsw"}</p> <p><b>Active State:</b> High</p>

## 5.5 External Data FIFO (DFIFO) RAM Interface Signals



This interface connects an external SPRAM as the Data FIFO RAM of the DWC\_otg core. Features supported on this interface include:

- Usable for both receive and transmit FIFOs
- RAM size (depth) selectable with coreConsultant
- SPRAM runs on the AHB clock
- Synchronous read and write operation

For integration information, see the "PFC-to-SPRAM Interface" section.

**Table 5-5 External Data FIFO (DFIFO) RAM Interface Signals**

Port Name	I/O	Description
<code>dfifo_h_rdata[34:0]</code>	I	<p>Read Data. Provides data read from the RAM address specified by <code>dfifo_h_addr</code>.</p> <p><b>Note:</b> This bit contains both data and status information. This bit is registered only for the data path. For the status path, it is not registered in the core.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ Bits [34:32]: Byte-enable and packet delimiter fields</li> <li>■ Bits [31:0]: Read data</li> </ul> <p><b>Exists:</b> Always  <b>Synchronous To:</b> <code>hclk_gated</code>  <b>Registered:</b> No  <b>Power Domain:</b> {OTG_EN_PWR0OPT==0 ? "Vdd" : "Vsw"}  <b>Active State:</b> High</p>
<code>hclk_gated</code>	O	<p>Data FIFO Clock. Provides gated AHB Clock to Data FIFO RAM.</p> <p><b>Exists:</b> Always  <b>Synchronous To:</b> None  <b>Registered:</b> No  <b>Power Domain:</b> {OTG_EN_PWR0OPT==1 ? "Vdd" : OTG_EN_PWR0OPT==2 ? "Vsw" : OTG_EN_PWR0OPT==3 ? "Vdd" : "Vdd"}  <b>Active State:</b> N/A</p>

**Table 5-5 External Data FIFO (DFIFO) RAM Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
dfifo_h_wr_n	O	<p>Write Enable. Qualifies dfifo_h_wdata and dfifo_h_addr</p> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> hclk,hclk_gated</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vsw" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> Low</p>
dfifo_h_ce_n	O	<p>Chip Enable. Indicates that the RAM is being accessed either for a read or a write.</p> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> hclk,hclk_gated</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vsw" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> Low</p>
dfifo_h_wdata[34:0]	O	<p>Write Data. Provides data to be written to the RAM address specified by dfifo_h_addr.</p> <p>Data is written to the RAM the rising edge of the AHB clock when dfifo_h_wr_n is sampled asserted.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ Bits [34:32]: Byte enable and packet delimiter</li> <li>■ Bits [31:0]: Write data</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> hclk,hclk_gated</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vsw" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-5 External Data FIFO (DFIFO) RAM Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
dfifo_h_addr[(OTG_DF_AWID-1):0]	O	<p>Read/Write Address. Provides the address for a read or write operation.</p> <p><b>Note:</b> The coreConsultant tool automatically calculates the address bus size using the Total Data FIFO RAM Depth (32 to 32,768) parameter. The formula is: <math>width = \log_2(OTG\_DFIFO\_DEPTH)</math></p> <p><b>Exists:</b> Always  <b>Synchronous To:</b> hclk,hclk_gated  <b>Registered:</b> No  <b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vsw" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}  <b>Active State:</b> High</p>

## 5.6 UTMI+ Parallel Interface Signals

utmi_txready - utmi_datain - utmi_rxvalidh - utmi_rxvalid - utmi_rxactive - utmi_rxerror - utmi_linenstate - utmi_hostdisconnect - utmifs_rx_dp - utmifs_rx_dm - utmifs_rx_rcv - utmifs_rx_se0 -	- utmi_dataout - utmi_txvalidh - utmi_txvalid - utmi_opmode - utmi_suspend_n - utmi_termselect - utmi_xcvrselect - utmi_word_if - utmi_fsls_low_power - utmi_fslsserialmode - utmifs_tx_enable_n - utmifs_tx_dat - utmifs_tx_dp - utmifs_tx_dm - utmifs_tx_se0 - utmifs_dp_rpu1_en_n - utmifs_dp_rpu2_en_n - utmifs_dm_pullup_en - utmifs_fs_edge_sel - utmi_fsdirection_oe
---	--

**Table 5-6 UTMI+ Parallel Interface Signals**

Port Name	I/O	Description
utmi_txready	I	<p>UTMI Transmit Data Ready. Indicates that the PHY accepted the current transmit data and is ready for the next packet the utmi_dataout bus.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT==3 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : ((OTG_HSPHY_DWIDTH!=0 &amp;&amp; (OTG_MODE!=5 &amp;&amp; OTG_MODE!=6)) ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : "phy_clk,pmu_clk") : "phy_clk,pmu_clk")) : ((OTG_MODE!=5 &amp;&amp; OTG_MODE!=6) ? (OTG_HSPHY_DWIDTH!=0 ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "phy_clk") : "phy_clk") : "phy_clk")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
utmi_datain[15:0]	I	<p>UTMI Receive Data. 8- or 16-bit data received from the PHY. The low and high bytes are asserted valid by the utmi_rxvalid and utmi_rxvalidh signals, respectively.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> utmi_clk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

Port Name	I/O	Description
utmi_rxvalidh	I	<p>UTMI Receive Data Valid, High Byte. Indicates that utmi_datain[15:8] contains valid data.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT==3 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : "utmifs_clk48,phy_clk,pmu_clk")) : (OTG_HSPHY_DWIDTH !=0 ? "phy_clk,pmu_clk" : "pmu_clk")) : (OTG_HSPHY_DWIDTH !=0 ? "phy_clk" : "None")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
utmi_rxvalid	I	<p>UTMI Receive Data Valid, Low Byte. Indicates that utmi_datain[7:0] contains valid data.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT==3 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" : "utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : "utmifs_clk48,phy_clk,pmu_clk")) : (OTG_FSPHY_INTERFACE==2 ? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk") : "phy_clk")</p> <p><b>Registered:</b> (OTG_FSPHY_INTERFACE == 2) ? "Yes": "No"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmi_rxactive	I	<p>UTMI Receive Active. Indicates that the PHY has detected SYNC and is active. This signal is deasserted after a bit stuff error or when an EOP is detected.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT==3 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : "phy_clk,pmu_clk"): (OTG_FSPHY_INTERFACE==2 ? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk") : "phy_clk")</p> <p><b>Registered:</b> (OTG_FSPHY_INTERFACE == 2) ? "Yes": "No"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
utmi_rxerror	I	<p>UTMI Receive Error. Indicates that the PHY has detected a receive error.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT==3 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : "phy_clk,pmu_clk"): (OTG_FSPHY_INTERFACE==2 ? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk") : "phy_clk")</p> <p><b>Registered:</b> (OTG_FSPHY_INTERFACE == 2) ? "Yes": "No"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

Port Name	I/O	Description
utmi_linestate[1:0]	I	<p>Line State Indicator. Indicates the current state of the two USB data signals:</p> <ul style="list-style-type: none"> <li>■ D+ (utmi_line_state[0]).</li> <li>■ D- (utmi_line_state[1]). This signal is combinational when the clock is not available (in Suspend state), but otherwise is a registered signal.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 2'b00: SE0</li> <li>■ 2'b01: J</li> <li>■ 2'b10: K</li> <li>■ 2'b11: SE1</li> </ul> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWRROPT==2)    (OTG_EN_PWRROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWRROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ? (OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : "phy_clk,pmu_clk") : (OTG_EN_PWRROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ? (OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : (OTG_ENABLE_LPM==1 ? (OTG_FSPHY_INTERFACE==0 ? (OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "phy_clk"))</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
utmi_dataout[15:0]	O	<p>UTMI Data Output Bus. 8- or 16-bit data transmitted to the PHY. The low and high bytes are asserted valid by utmi_rxvalid and utmi_rxvalidh, respectively.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> utmi_clk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

Port Name	I/O	Description
utmi_txvalidh	O	<p>UTMI Transmit Data Valid, High Byte. Indicates that utmi_dataout[15:8] contains valid data.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT==3 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : (OTG_HSPHY_DWIDTH !=0 ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : "phy_clk,pmu_clk") : (OTG_HSPHY_DWIDTH !=0 ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "phy_clk") : "None")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

Port Name	I/O	Description
utmi_txvalid	O	<p>UTMI Transmit Data Valid, Low Byte. Indicates that utmi_dataout[7:0] contains valid data.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_EN_ADPSUPPORT==1)) ? (OTG_EN_PWROPT==3 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : "phy_clk,pmu_clk")) : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "phy_clk")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

Port Name	I/O	Description
utmi_opmode[1:0]	O	<p>UTMI Operating Mode. The DWC_otg core drives this signal to select the UTMI mode.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 2'b00: Normal operation</li> <li>■ 2'b01: Non-driving</li> <li>■ 2'b10: Disable bit stuffing and NRZI encoding No other values are supported.</li> </ul> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_EN_ADSP_SUPPORT==1)) ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : "phy_clk,pmu_clk"): (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : (OTG_SYNC_RESET_TYPE==1 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "phy_clk"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

Port Name	I/O	Description
utmi_suspend_n	O	<p>UTMI Suspend. Places the transceiver/PHY in Suspend mode, drawing minimal power. The FS transceiver interface uses this UTMI+ PHY signal.</p> <p>In this mode:</p> <ul style="list-style-type: none"> <li>■ The transceiver/PHY shuts down all internal circuits that are not necessary for suspend/resume.</li> <li>■ The PHY disables the clock.</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT==0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk,phy_clk" :"utmi_clk,ulpi_clk,pmu_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk,phy_clk")) : (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk" :"utmi_clk,ulpi_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk")) : "utmifs_clk48,pmu_clk")))) : OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" :"utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" :"utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48")</p> <p><b>Registered:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? "No" : "Yes"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> Low</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

Port Name	I/O	Description
utmi_termselect	O	<p>FS Shared-Pin Interface:.. While suspended, used as the UTMI termination select signal, utmi_termselect.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT==3 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :"phy_clk,pmu_clk")) : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : (OTG_SYNC_RESET_TYPE==1 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "phy_clk"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

Port Name	I/O	Description
utmi_xcvrselect[1:0]	O	<p>Transceiver Select. Selects a HS, FS, or LS transceiver the PHY.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 2'b00: HS transceiver enabled</li> <li>■ 2'b01: FS transceiver enabled</li> <li>■ 2'b10: LS transceiver enabled</li> <li>■ 2'b11: Send a LS packet a FS bus or receive a LS packet. If xcvrselect is 2'b11, the transceiver sends a preamble packet at FS before sending the LS packet.</li> </ul> <p>In receive mode, the transceiver waits to receive an LS packet with the LS transceiver enabled. The transceiver must send all data (both FS preamble packet and the LS data) with FS signaling (fast rise and fall times, and opposite polarity)</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADAP_SUPPORT==1)) ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : "phy_clk,pmu_clk")) : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk")) : (OTG_SYNC_RESET_TYPE==1 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "phy_clk"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmi_word_if	O	<p>UTMI Data Bus Width and Clock Select. Indicates whether the PHY is operating in 8- or 16-bit mode.</p> <p><b>Note:</b> This is not an actual UTMI+ Level 3 signal.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: 8-bit interface</li> <li>■ 1'b1: 16-bit interface</li> </ul> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT==3 ? "utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_DWIDTH !=0 ? "pmu_clk" : "None")) : "None"</p> <p><b>Registered:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P SUPPORT ==1)    (OTG_HSPHY_DWIDTH != 2)) ? "No" : "Yes"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
utmi_hostdisconnect	I	<p>Peripheral Disconnect Indicator to Host. Indicates that the USB transceiver has detected a disconnect condition the cable.</p> <ul style="list-style-type: none"> <li>■ This signal is valid only when utmiotg_dppulldown and utmiotg_dmpulldown are sampled asserted.</li> <li>■ When utmiotg_dppulldown and utmiotg_dmpulldown are not sampled asserted, then the behavior of utmi_hostdisconnect is undefined. Connection:           <ul style="list-style-type: none"> <li>■ No peripheral connected: signal remains asserted.</li> <li>■ Peripheral connected: signal de-asserted.</li> </ul> </li> </ul> <p><b>Exists:</b> ((OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3) &amp;&amp; OTG_MODE !=3 &amp;&amp; OTG_MODE !=4)</p> <p><b>Synchronous To:</b> OTG_EN_ACG!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "phy_clk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vdd" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

Port Name	I/O	Description
utmi_fsls_low_power	O	<p>PHY Low-Power Clock Select. Selects either 480- or 48-MHz Low-Power mode for the PHY. Typically in FS and LS modes, the PHY can operate a 48-MHz clock to save power.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: 480-MHz Internal PLL clock</li> <li>■ 1'b1: 48-MHz External clock Clocks:</li> <li>■ Using a 48-MHz clock, the UTMI interface operates at 48 MHz in FS mode.</li> <li>■ Using a 480-MHz clock, the UTMI interface operates at either 60 MHz for 8-bit data mode, or 30 MHz for 16-bit data mode.</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT==3 ? "utmi_clk,pmu_clk" : ((OTG_HSPHY_INTERFACE == 0)    (OTG_HSPHY_INTERFACE == 2)) ? "None" : "pmu_clk") : "None"</p> <p><b>Registered:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P SUPPORT == 1)    (OTG_HSPHY_INTERFACE == 0)    (OTG_HSPHY_INTERFACE == 2)) ? "No" : "Yes"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
utmi_fslsserialmode	O	<p>PHY Interface Mode Select. Indicates which PHY interface is being used to transfer the FS packets.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: FS packets are transferred through the parallel interface</li> <li>■ 1'b1: FS packets are transferred through the serial interface</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT==3 ? "utmi_clk,phy_clk,pmu_clk" : (OTG_FSPHY_INTERFACE == 0 ? "None" : "pmu_clk")) : "None"</p> <p><b>Registered:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P SUPPORT == 1)    (OTG_FSPHY_INTERFACE == 0)    (OTG_HSPHY_INTERFACE == 0)) ? "No" : "Yes"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmifs_tx_enable_n	O	<p>Output Enable Data valid output enable for the following signals:</p> <ul style="list-style-type: none"> <li>■ utmifs_tx_dp</li> <li>■ utmifs_tx_dm</li> <li>■ utmifs_tx_se0</li> <li>■ utmifs_tx_dat</li> </ul> <p>For the FS Shared-Pin Interface, Bit 0 connects to utmifs_tx_enable_n. For the ULPI interface, utmifs_tx_enable_n uses ulpi_dataout[0].</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE == 1)  <b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P SUPPORT ==1)) ? (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,phy_clk,pmu_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,phy_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,phy_clk")) : "utmifs_clk48,phy_clk"))  <b>Registered:</b> No  <b>Power Domain:</b> Vdd  <b>Active State:</b> Low</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmifs_tx_dat	O	<p>Transmit Differential Serial Data Indicates that D+ /D. lines must be driven with J or K states. It is valid when utmi_tx_enable_n is asserted (low).</p> <p><b>Note:</b> This signal must be used only when utmifs_tx_dp and utmifs_tx_dm are not used.</p> <p><b>Values when operating in FS:</b></p> <ul style="list-style-type: none"> <li>■ 1'b1: J state (D+ = 1'b1; D. = 1'b0)</li> <li>■ 1'b0: K state (D+ = 1'b0; D. = 1'b1)</li> </ul> <p><b>Values when operating in LS:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: J state (D+ = 1'b0; D. = 1'b1)</li> <li>■ 1'b1: K state (D+ = 1'b1; D. = 1'b0)</li> </ul> <p>For the FS Shared-Pin Interface, Bit 1 connects to utmifs_tx_dat. For the ULPI interface, utmifs_tx_dat uses ulpi_dataout[1].</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE == 1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT ==1)) ? (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,pmu_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,pmu_clk")): "utmifs_clk48,pmu_clk") : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk" : "utmifs_clk48,utmi_clk,ulpi_clk")): "utmifs_clk48")</p> <p><b>Registered:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT ==1)) ? "No":"Yes"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmifs_tx_dp	O	<p>Single-ended D+ Line Single-ended D+ line driver. It is valid only when utmi_tx_enable_n is asserted (low).</p> <p><b>Note:</b> This signal must be used only when utmifs_tx_se0 and utmifs_tx_dat are not used.</p> <p>For the FS Shared-Pin Interface, Bit 7 connects to utmifs_tx_dp. For the ULPI interface, utmifs_tx_dp uses ulpi_dataout[7].</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE == 1)  <b>Synchronous To:</b> ((OTG_EN_PWRLOPT == 2)    (OTG_EN_PWRLOPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? (OTG_HSPHY_INTERFACE != 0) : (OTG_HSPHY_INTERFACE == 1 ? "utmifs_clk48,utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE == 2 ? "utmifs_clk48,ulpi_clk,pmu_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,pmu_clk")) : "utmifs_clk48,pmu_clk" : (OTG_HSPHY_INTERFACE != 0) : (OTG_HSPHY_INTERFACE == 1 ? "utmifs_clk48,utmi_clk" : (OTG_HSPHY_INTERFACE == 2 ? "utmifs_clk48,ulpi_clk" : "utmifs_clk48,utmi_clk,ulpi_clk")) : "utmifs_clk48")  <b>Registered:</b> ((OTG_EN_PWRLOPT == 2)    (OTG_EN_PWRLOPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? "No" : "Yes"  <b>Power Domain:</b> Vdd  <b>Active State:</b> High</p>
utmifs_tx_dm	O	<p>Single-ended D- Line Single-ended D- line driver. It is valid only when utmi_tx_enable_n is asserted (low).</p> <p><b>Note:</b> This signal must be used only when utmifs_tx_se0 and utmifs_tx_dat are not used.</p> <p>For the FS Shared-Pin Interface, utmifs_tx_dm uses utmi_dataout[5].</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE == 1)  <b>Synchronous To:</b> ((OTG_EN_PWRLOPT == 2)    (OTG_EN_PWRLOPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? (OTG_HSPHY_INTERFACE != 0) : (OTG_HSPHY_INTERFACE == 1 ? "utmifs_clk48,utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE == 2 ? "utmifs_clk48,ulpi_clk,pmu_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,pmu_clk")) : "utmifs_clk48,pmu_clk" : (OTG_HSPHY_INTERFACE != 0) : (OTG_HSPHY_INTERFACE == 1 ? "utmifs_clk48,utmi_clk" : (OTG_HSPHY_INTERFACE == 2 ? "utmifs_clk48,ulpi_clk" : "utmifs_clk48,utmi_clk,ulpi_clk")) : "utmifs_clk48")  <b>Registered:</b> ((OTG_EN_PWRLOPT == 2)    (OTG_EN_PWRLOPT == 3)    (OTG_AD_P_SUPPORT == 1)) ? "No" : "Yes"  <b>Power Domain:</b> Vdd  <b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmifs_tx_se0	O	<p>Transmit Serial SE0 Assertion Indicates that both D+ and D- lines must be driven with SE0, independent of utmifs_tx_dat. It is valid when utmi_tx_enable_n is asserted (low).</p> <p><b>Note:</b> This signal must be used only when utmifs_tx_dp and utmifs_tx_dm are not used.</p> <p>For the FS Shared-Pin Interface, utmifs_tx_se0 uses ulpi_dataout[2].</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE == 1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,pmu_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,pmu_clk")) : "utmifs_clk48,pmu_clk") : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk" : "utmifs_clk48,utmi_clk,ulpi_clk")) : "utmifs_clk48")</p> <p><b>Registered:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P SUPPORT ==1)) ? "No":"Yes"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
utmifs_rx_dp	I	<p>FS/LS Single-Ended D+ Line Single-ended D+ signal from the transceiver.</p> <p>For the FS Shared-Pin Interface, Bit 4 connects to utmifs_rx_dp. For the ULPI Interface, this signal uses ulpi_datain[4].</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE == 1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,phy_clk,pmu_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : "utmifs_clk48,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,phy_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,phy_clk")) : "utmifs_clk48,phy_clk")</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmifs_rx_dm	I	<p>FS/LS Single-Ended D- Line Single-ended D- signal from the transceiver.FS Shared-Pin Interface. For the FS Shared-Pin Interface, Bit 5 connects to utmifs_rx_dm. For the ULPI interface, utmifs_rx_dm uses ulpi_datain[5].</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE == 1)  <b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P_SUPPORT==1)) ? (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,phy_clk,pmu_clk" : _utmifs_clk48,utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : "utmifs_clk48,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,phy_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,phy_clk")) : "utmifs_clk48,phy_clk")</p> <p><b>Registered:</b> Yes  <b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
utmifs_rx_rcv	I	<p>FS/LS Transceiver Differential Output Differential receive data from D+ and D- lines. For the FS Shared-Pin Interface, Bit 6 connects to utmifs_rx_rcv. For the ULPI interface, utmifs_rx_rcv uses ulpi_datain[6].</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE == 1)  <b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P_SUPPORT==1)) ? (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,phy_clk,pmu_clk" : _utmifs_clk48,utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : "utmifs_clk48,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,phy_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,phy_clk")) : "utmifs_clk48,phy_clk")</p> <p><b>Registered:</b> Yes  <b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmifs_rx_se0	I	<p>Serial SE0 Assertion Received. Indicates that the transceiver received SE0 (Single-Ended Zero) both D+ and D- lines.</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE == 1    OTG_FSPHY_INTERFACE == 2    OTG_FSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADAP_SUPPORT==1)) ? (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,phy_clk,pmu_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : "utmifs_clk48,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmifs_clk48,utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"utmifs_clk48,ulpi_clk,phy_clk" : "utmifs_clk48,utmi_clk,ulpi_clk,phy_clk")) : "utmifs_clk48,phy_clk")</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

Port Name	I/O	Description
utmifs_dp_rpu1_en_n	O	<p>D+ Pull-up Resistors Enable, SW1.</p> <ul style="list-style-type: none"> <li>■ In FS mode of operation, when asserted, it signals the FS transceiver to enable two serially-connected pull-up resistors on the D+ line.</li> <li>■ In LS mode of operation, when asserted, it signals the LS transceiver to enable two serially-connected pull-up resistors on the D- line.</li> </ul> <p><b>Note:</b> This signal is used to control the SW1 switch. Because the switch is applicable to D+ line in FS mode of operation and D- line in LS mode of operation, external connection must be done accordingly to D+ or D- line based on the speed of operation.</p> <p>This is shown in the <i>USB Engineering Change Notice (ECN): Pull-up/pull-down resistors</i>, for the USB 2.0 Specification. This signal controls SW1 in the ECN.</p> <p>For the FS Shared-Pin Interface, utmifs_dp_rpu1_en_n uses utmi_dataout[0].</p> <p><b>Exists:</b> ((OTG_FSPHY_INTERFACE == 1    OTG_FSPHY_INTERFACE == 3) &amp;&amp; OTG_I2C_INTERFACE != 1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADP_SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" : "utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" : "utmifs_clk48,phy_clk,pmu_clk")) : ((OTG_MODE==6 &amp;&amp; OTG_I2C_INTERFACE==0) ? "phy_clk" : (OTG_FSPHY_INTERFACE==0) ? (OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk"))))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> Low</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

Port Name	I/O	Description
utmifs_dp_rpu2_en_n	O	<p>D+ Pull-up Resistor Enable in FS mode and D- Pull-up Resistor Enable in LS mode.</p> <p>This signal controls SW2 in the <i>USB Engineering Change Notice (ECN): Pull-up/pull-down resistors</i>, for the USB 2.0 Specification. You can avoid using this signal by using a discrete pull-up resistor instead. For the FS Shared-Pin Interface, utmifs_dp_rpu2_en_n uses utmi_dataout[1].</p> <p><b>Exists:</b> ((OTG_FSPHY_INTERFACE == 1    OTG_FSPHY_INTERFACE == 3) &amp;&amp; OTG_I2C_INTERFACE != 1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" : "utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" : "utmifs_clk48,phy_clk,pmu_clk")) : ((OTG_MODE==6 &amp;&amp; OTG_I2C_INTERFACE==0) ? "phy_clk" : (OTG_FSPHY_INTERFACE==0) ? (OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk"))))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> Low</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmifs_dm_pullup_en	O	<p>D- Pull-up Resistor Enable.</p> <p>This signal is used in LS mode of operation to enable pull-up on D-line if variable RPU is not implemented on LS transceiver.</p> <p>When asserted, the FS/LS transceiver enables the pull-up resistor on the D- line for LS mode of operation. For the FS Shared-Pin Interface, utmifs_dm_pullup_en uses utmi_dataout[2].</p> <p><b>Exists:</b> ((OTG_FSPHY_INTERFACE == 1    OTG_FSPHY_INTERFACE == 3) &amp;&amp; OTG_I2C_INTERFACE != 1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADSP_SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" : "utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" : "utmifs_clk48,phy_clk,pmu_clk")) : ((OTG_MODE==6 &amp;&amp; OTG_I2C_INTERFACE==0) ? "phy_clk" : (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk")))))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-6 UTMI+ Parallel Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmifs_fs_edge_sel	O	<p>Speed Select. Indicates whether the device is full speed or low speed and selects the slew rate of the D+ and D- line drivers.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Low speed selected</li> <li>■ 1'b1: Full speed selected</li> </ul> <p>For the FS Shared-Pin Interface, utmifs_fs_edge_sel uses utmi_dataout[3].</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE == 1    OTG_FSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P_SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" : "utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" : "utmifs_clk48,phy_clk,pmu_clk")) : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk")) : "phy_clk")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
utmi_fsdirection_oe	O	<p>UTMI FS (Pin Shared) Data Output Enable. For the FS Shared-Pin Interface, UTMI FS data output enable is driven by the core to the pad enables of 'dat' and 'se0' drivers in FS Pin Shared mode.</p> <p><b>Note:</b> This signal is not defined in the UTMI+ specification.</p> <p><b>Exists:</b> (((OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3) &amp;&amp; OTG_FSPHY_INTERFACE == 2))</p> <p><b>Synchronous To:</b> utmifs_clk48,utmi_clk,phy_clk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

## 5.7 UTMI+ PMU Interface Signals

utmi\_hostdisconnect\_pmu -  
 utmiotg\_iddig\_pmu -  
 utmisrp\_bvalid\_pmu -  
 utmiotg\_vbusvalid\_pmu -  
 utmisrp\_sessend\_pmu -

**Table 5-7** UTMI+ PMU Interface Signals

Port Name	I/O	Description
utmi_hostdisconnect_pmu	I	<p>Peripheral Disconnect Indicator to Host for PMU. Indicates that the USB transceiver has detected a disconnect condition the cable.</p> <ul style="list-style-type: none"> <li>■ This signal is valid only when utmiotg_dppulldown and utmiotg_dmpulldown are sampled asserted.</li> <li>■ When utmiotg_dppulldown and utmiotg_dmpulldown are not sampled asserted, then the behavior of utmi_hostdisconnect is undefined. Connection:           <ul style="list-style-type: none"> <li>■ No peripheral connected: signal remains asserted.</li> <li>■ Peripheral connected: signal de-asserted.</li> </ul> </li> </ul> <p><b>Exists:</b> ((OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3) &amp;&amp; (OTG_EN_PWROPT == 2    OTG_EN_PWROPT == 3    OTG_AD_PDP_SUPPORT == 1) &amp;&amp; OTG_MODE !=3 &amp;&amp; OTG_MODE !=4)</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-7 UTMI+ PMU Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmiotg_iddig_pmu	I	<p>Mini A/B Plug Indicator for PMU. Indicates whether the connected plug is mini-A or mini-B. Valid only when utmiotg_idpullup is sampled asserted.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Mini-A connected</li> <li>■ 1'b1: Mini-B connected</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>■ This signal is not present when OTG_BC_SUPPORT=1.</li> <li>■ This signal must always be tied to 1'b1 after hreset_n is de-asserted.</li> </ul> <p><b>Exists:</b> ((OTG_EN_PWRLOPT == 2    OTG_EN_PWRLOPT == 3    OTG_ADSP_SUPPORT == 1) &amp;&amp; OTG_BC_SUPPORT == 0 &amp;&amp; ((OTG_MODE == 0    OTG_MODE == 1    OTG_MODE == 2    OTG_MODE == 5) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> pmu_clk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
utmisrp_bvalid_pmu	I	<p>B-Peripheral Session Valid for PMU. Indicates if the voltage Vbus is at a valid level for a B-peripheral session. The comparator thresholds are:</p> <p>OTG Revision 1.3 and Revision 2.0 Values:</p> <ul style="list-style-type: none"> <li>■ 1'b0: Vbus &lt; 0.8 V</li> <li>■ 1'b1: Vbus &gt;= 4 V</li> </ul> <p>Device Mode:</p> <ul style="list-style-type: none"> <li>■ Asserting this signal In Device mode sets the Session Request / New Session Detected Interrupt bit in the Interrupt Register (GINTSTS) (GINTSTS.SessReqInt).</li> <li>■ De-asserting this signal sets the Session End Detected bit in the Interrupt Register (GOTGINT) (GOTGINT.SesEndDet).</li> </ul> <p><b>Exists:</b> ((OTG_EN_PWRLOPT == 2    OTG_EN_PWRLOPT == 3    OTG_ADSP_SUPPORT == 1) &amp;&amp; (OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> pmu_clk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-7 UTMI+ PMU Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmiotg_vbusvalid_pmu	I	<p>Vbus Valid for PMU. Indicates if the voltage Vbus is valid for A/B-device/peripheral operation. The comparator thresholds are:</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Vbus &lt; 4.4 V</li> <li>■ 1'b1: Vbus &gt; 4.75 V</li> </ul> <p><b>Exists:</b> ((OTG_EN_PWROPT == 2    OTG_EN_PWROPT == 3    OTG_AD_P_SUPPORT == 1) &amp;&amp; (OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
utmisrp_sessend_pmu	I	<p>B-Device Session End for PMU. Indicates if the voltage Vbus is below the B-device Session End threshold. The comparator thresholds are:</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Vbus &gt; 0.8 V</li> <li>■ 1'b1: Vbus &lt; 0.2 V</li> </ul> <p><b>Exists:</b> ((OTG_EN_PWROPT == 2    OTG_EN_PWROPT == 3    OTG_AD_P_SUPPORT == 1) &amp;&amp; (OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

## 5.8 UTMI+ OTG Interface Signals

utmiotg_iddig -	- utmiotg_idpullup
utmiotg_avalid -	- utmiotg_dppulldown
utmisrp_bvalid -	- utmiotg_dmpulldown
utmiotg_vbusvalid -	- utmiotg_drvvbus
utmisrp_sessend -	- utmisrp_chrgvbus - utmisrp_dischrgvbus

For additional information on the following signals, see the "Overriding Control of PHY Voltage Valid and ID Input Signals" section in the databook.

- utmiotg\_iddig
- utmiotg\_avalid
- utmisrp\_bvalid
- utmiotg\_vbusvalid

**Table 5-8 UTMI+ OTG Interface Signals**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmiotg_iddig	I	<p>Mini A/B Plug Indicator. Indicates whether the connected plug is mini-A or mini-B. Valid only when utmiotg_idpullup is sampled asserted.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Mini-A connected</li> <li>■ 1'b1: Mini-B connected</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>■ This signal is not present when OTG_BC_SUPPORT=1. This needs to be derived as outlined in 'Batter Charger Support' in the DesignWare Cores USB 2.0 Hi-Speed On-The-Go (OTG) User Guide.</li> <li>■ This signal should always be tied to 1'b1 after hreset_n is de-asserted.</li> <li>■ This signal is also present in SRP capable Host mode.</li> </ul> <p>For more information, see the "Overriding Control of PHY Voltage Valid and ID Input Signals" section in the databook.</p> <p><b>Exists:</b> (OTG_BC_SUPPORT == 0 &amp;&amp; ((OTG_MODE == 0    OTG_MODE == 1    OTG_MODE == 2    OTG_MODE == 5) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1))))</p> <p><b>Synchronous To:</b> OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0 ? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")): "utmifs_clk48") : "phy_clk"</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vdd" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-8 UTMI+ OTG Interface Signals (Continued)**

Port Name	I/O	Description
utmiotg_avalid	I	<p>A-Peripheral Session Valid. Indicates if the voltage Vbus is at a valid level for an A-peripheral session. The comparator thresholds are:</p> <p>OTG Revision 1.3 Values:</p> <ul style="list-style-type: none"> <li>■ 1'b0: Vbus &lt; 0.8 V</li> <li>■ 1'b1: Vbus = 0.8 V to 2.0 V</li> </ul> <p>OTG Revision 2.0 Values:</p> <ul style="list-style-type: none"> <li>■ 1'b0: Vbus &lt; 0.8 V</li> <li>■ 1'b1: Vbus = 0.8 V to 4 V</li> </ul> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>■ In OTG2.0 mode, the utmiotg_avalid and utmiotg_bvalid signals must be connected to VOTG_SESS_VLD from the PHY.</li> <li>■ For HSIC configuration, the valid input signal to the controller cannot be connected from the HSIC PHY because there is no Vbus pin present in the HSIC PHY. Therefore, if the valid input needs to be connected to a proper value when operating in HSIC mode, the valid input value can be alternatively overridden using the software control bits in the controller.</li> </ul> <p>For more information, see the "Overriding Control of PHY Voltage Valid and ID Input Signals" section in the databook.</p> <p><b>Exists:</b> ((OTG_MODE !=2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")): "utmifs_clk48")) : "phy_clk"</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vdd" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-8 UTMI+ OTG Interface Signals (Continued)**

Port Name	I/O	Description
utmisrp_bvalid	I	<p>B-Peripheral Session Valid. Indicates if the voltage Vbus is at a valid level for a B-peripheral session. The comparator thresholds are: OTG Revision 1.3 and Revision 2.0 Values:</p> <ul style="list-style-type: none"> <li>■ 1'b0: Vbus &lt; 0.8 V</li> <li>■ 1'b1: Vbus = 0.8 V to 4 V</li> </ul> <p>Device Mode:</p> <ul style="list-style-type: none"> <li>■ Asserting this signal In Device mode sets the Session Request/New Session Detected Interrupt bit in the Interrupt Register (GINTSTS) (GINTSTS.SessReqInt).</li> <li>■ Deasserting this signal sets the Session End Detected bit in the Interrupt Register (GOTGINT) (GOTGINT.SesEndDet).</li> </ul> <p><b>Note:</b></p> <p>For HSIC configuration, the bvalid input to controller cannot be connected from the HSIC PHY because there is no Vbus pin present in the HSIC PHY. Therefore, if the bvalid input needs to be connected to a proper value when operating in HSIC mode, the bvalid input value can be alternatively overridden using the software control bits in the controller. For more information, see the "Overriding Control of PHY Voltage Valid and ID Input Signals" section in the databook.</p> <p><b>Exists:</b> (((OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")): "utmifs_clk48") : "phy_clk"</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vdd" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-8 UTMI+ OTG Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmiotg_vbusvalid	I	<p>Vbus Valid. Indicates if the voltage Vbus is valid for A/B-device/peripheral operation. The comparator thresholds are:</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Vbus &lt; 4.4 V</li> <li>■ 1'b1: Vbus &gt; 4.75 V</li> </ul> <p>For more information, see the "Overriding Control of PHY Voltage Valid and ID Input Signals" section in the databook.</p> <p><b>Exists:</b> ((OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")): "utmifs_clk48")) : "phy_clk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vdd" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-8 UTMI+ OTG Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmisrp_sessend	I	<p>B-Device Session End. Indicates if the voltage Vbus is below the B-device Session End threshold. The comparator thresholds are:</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Vbus &gt; 0.8 V</li> <li>■ 1'b1: Vbus &lt; 0.2 V</li> </ul> <p><b>Exists:</b> ((OTG_MODE !=2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")): "utmifs_clk48")) : "phy_clk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vdd" : "Vdd"}</p> <p><b>Active State:</b> High</p>
utmiotg_idpullup	O	<p>Analog ID Input Sample Enable. Enables sampling the analog ID line.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: ID pin sampling disabled</li> <li>■ 1'b1: ID pin sampling enabled</li> </ul> <p><b>Exists:</b> (OTG_MODE == 0 &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-8 UTMI+ OTG Interface Signals (Continued)**

Port Name	I/O	Description
utmiotg_dppulldown	O	<p>D+ Pull-down Resistor Enable. Enables the 15 KW pull-down resistor on the D+ line.</p> <p><b>Exists:</b> (OTG_MODE == 0 &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADSP_SUPPORT==1)) ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :(OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :(OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmifs_clk48,phy_clk,pmu_clk")) : "phy_clk,pmu_clk")) : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :(OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :(OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk")) : "phy_clk")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-8 UTMI+ OTG Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmiotg_dmpulldown	O	<p>D' Pull-down Resistor Enable. Enables the 15 KW pull-down resistor on the D' line.</p> <p><b>Exists:</b> (OTG_MODE == 0 &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" :(OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" :(OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmifs_clk48,phy_clk,pmu_clk")) : "phy_clk,pmu_clk")) : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" :(OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :(OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk")) : "phy_clk")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-8 UTMI+ OTG Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmiotg_drvvbus	O	<p>Drive Vbus. Enables driving Vbus to 5 V.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Do not drive Vbus</li> <li>■ 1'b1: Drive Vbus</li> </ul> <p><b>Exists:</b> (((OTG_MODE !=2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))    ((OTG_TYPEC_MODE == 1) &amp;&amp; (OTG_MODE !=3    OTG_MODE != 4) ))</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_EN_ADPSUPPORT==1)) ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk,phy_clk" : "utmi_clk,ulpi_clk,pmu_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk,phy_clk" : "utmifs_clk48,pmu_clk,phy_clk")) : "phy_clk,pmu_clk")) : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk")) : (OTG_MODE!=3 ? (OTG_TYPEC_MODE==1 ? "phy_clk,cc_clk" : "phy_clk") : "None")))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-8 UTMI+ OTG Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmisrp_chrgvbus	O	<p>Vbus Input Charge Enable. Directs the PHY to charge Vbus.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Do not charge Vbus through a resistor.</li> <li>■ 1'b1: Charge Vbus through a resistor (must be active for at least 30 ms).</li> </ul> <p><b>Exists:</b> ((OTG_MODE !=2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" : "utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" : "utmifs_clk48,phy_clk,pmu_clk")) : "phy_clk,pmu_clk")) : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk")) : "phy_clk")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-8 UTMI+ OTG Interface Signals (Continued)**

Port Name	I/O	Description
utmisrp_dischrgvbus	O	<p>Vbus Input Discharge Enable. This signal directs the PHY to discharge Vbus.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Do not discharge Vbus through a resistor.</li> <li>■ 1'b1: Discharge Vbus through a resistor (The signal must be active for at least 50 ms).</li> </ul> <p><b>Exists:</b> ((OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3    (OTG_FSPHY_INTERFACE != 0 &amp;&amp; OTG_I2C_INTERFACE != 1)))</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_EN_ADPSUPPORT==1)) ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" : "utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")) : "utmifs_clk48,phy_clk,pmu_clk")) : (phy_clk,pmu_clk)): (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk")) : "phy_clk")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

## 5.9 UTMI+ Vendor Control Interface Signals



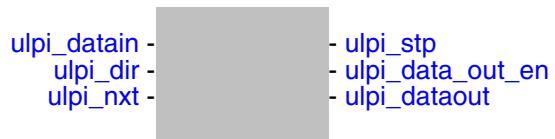
**Table 5-9** UTMI+ Vendor Control Interface Signals

Port Name	I/O	Description
utmi_vcontrolload_n	O	<p>Vendor Control Load. Loads the vendor control register in the PHY.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Load vendor control register</li> <li>■ 1'b1: No operation</li> </ul> <p><b>Exists:</b> ((OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3) &amp;&amp; OTG_VENDOR_CTL_INTERFACE==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_EN_PWR_CLAMP == 0) ? "phy_clk,pmu_clk" : (OTG_EN_PWR_CLAMP == 1) ? "utmi_clk,ulpi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "phy_clk" : "phy_clk")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> Low</p>

**Table 5-9 UTMI+ Vendor Control Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmi_vcontrol[3:0]	O	<p>Vendor Control. Vendor defined parallel output bus.</p> <p><b>Exists:</b> ((OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3) &amp;&amp; OTG_VENDOR_CTL_INTERFACE==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADAP_SUPPORT==1)) ? "pmu_clk" : (OTG_PWR_CLAMP == 0 ? (OTG_EN_PWROPT!=0 ?(OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" :"utmi_clk,ulpi_clk") :(OTG_HSPHY_INTERFACE!=0?(OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" :"utmi_clk,ulpi_clk,utmifs_clk48"): "utmifs_clk48")) : "None") : "None")</p> <p><b>Registered:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADAP_SUPPORT==1)) ? "No" : (OTG_PWR_CLAMP ==0 ? "No" : "Yes")</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
utmi_vstatus[7:0]	I	<p>Vendor Status. Vendor-defined parallel input bus.</p> <p><b>Exists:</b> ((OTG_HSPHY_INTERFACE == 1    OTG_HSPHY_INTERFACE == 3) &amp;&amp; OTG_VENDOR_CTL_INTERFACE==1)</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vdd" : "Vdd"}</p> <p><b>Active State:</b> High</p>

## 5.10 ULPI Interface Signals



**Table 5-10 ULPI Interface Signals**

Port Name	I/O	Description
ulpi_stp	O	<p>Stop Output Control. The DWC_otg core asserts this signal for one clock cycle to indicate the end of a USB transmit packet or a register write operation and, optionally, to stop packet reception. Assertion occurs after the last data byte is presented the bus.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADAP_SUPPORT==1)) ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ? (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk" : "ulpi_clk,utmi_clk,pmu_clk") : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk" : "ulpi_clk,utmi_clk,utmifs_clk48,pmu_clk") : "ulpi_clk,utmifs_clk48,pmu_clk")) : "ulpi_clk,phy_clk,pmu_clk") : (OTG_FSPHY_INTERFACE==0 ? (OTG_EN_PWROPT!=0 ? (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "ulpi_clk,utmi_clk") : "ulpi_clk,phy_clk") : (OTG_EN_PWROPT!=0 ? (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "ulpi_clk,utmi_clk,utmifs_clk48") : "ulpi_clk,utmifs_clk48")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "ulpi_clk,utmi_clk,utmifs_clk48,phy_clk") : "ulpi_clk,utmifs_clk48,phy_clk"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-10 ULPI Interface Signals (Continued)**

Port Name	I/O	Description
ulpi_data_out_en[7:0]	O	<p>ULPI FS (Pin Shared) Data Output Enable.  <b>Note:</b> This signal is not defined in the ULPI specification.          ULPI data output enable driven by the core to the pad enables of drivers in ULPI mode or FS Pin Shared mode.</p> <p><b>ULPI Interface:</b></p> <ul style="list-style-type: none"> <li>■ Bit 0 connects to the pad enables of ulpi_dataout[0].</li> <li>■ Bit 1 connects to the pad enables of ulpi_dataout[1].</li> <li>■ Bit 2 connects to the pad enables of ulpi_dataout[2].</li> <li>■ Bit 3 connects to the pad enables of ulpi_dataout[3].</li> <li>■ Bit 4 connects to the pad enables of ulpi_dataout[4].</li> <li>■ Bit 5 connects to the pad enables of ulpi_dataout[5].</li> <li>■ Bit 6 connects to the pad enables of ulpi_dataout[6].</li> <li>■ Bit 7 connects to the pad enables of ulpi_dataout[7].</li> </ul> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)  <b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0 ? "ulpi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2? "ulpi_clk,phy_clk,utmifs_clk48,pmu_clk" : "ulpi_clk,utmif_clk,utmifs_clk48,phy_clk,pmu_clk")) : (OTG_FSPHY_INTERFACE==0 ? "ulpi_clk" : (OTG_HSPHY_INTERFACE==2? "ulpi_clk,phy_clk,utmifs_clk48" : "ulpi_clk,utmif_clk,utmifs_clk48,phy_clk"))  <b>Registered:</b> No  <b>Power Domain:</b> Vdd  <b>Active State:</b> High</p>
ulpi_dataout[7:0]	O	<p>ULPI Data Output. ULPI data output bus driven by the DWC_otg core.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)  <b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE!=3 ? "ulpi_clk,pmu_clk" : "ulpi_clk,utmif_clk,utmifs_clk48,phy_clk,pmu_clk") : (OTG_FSPHY_INTERFACE==0 ? "ulpi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "ulpi_clk,utmif_clk,utmifs_clk48,phy_clk"))  <b>Registered:</b> No  <b>Power Domain:</b> Vdd  <b>Active State:</b> High</p>

**Table 5-10 ULPI Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
ulpi_datain[7:0]	I	<p>ULPI Data Input. ULPI data input to the core driven by the PHY.</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0 ? (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" : "ulpi_clk,utmi_clk,phy_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" : "ulpi_clk,utmi_clk,utmifs_clk48,phy_clk,pmu_clk")) : (OTG_FSPHY_INTERFACE==0 ? (OTG_EN_PWROPT!=0 ? (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "ulpi_clk,utmi_clk,phy_clk") : "ulpi_clk,phy_clk") : (OTG_EN_PWROPT!=0 ? (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "ulpi_clk,utmi_clk,utmifs_clk48,phy_clk") : "ulpi_clk,phy_clk"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-10 ULPI Interface Signals (Continued)**

Port Name	I/O	Description
ulpi_dir	I	<p>Data Bus Control. Controls the direction of the data bus.</p> <ul style="list-style-type: none"> <li>■ When the PHY has data to transfer to the DWC_otg core, it drives ulpi_dir high to take ownership of the bus.</li> <li>■ When the PHY has no data to transfer, it drives ulpi_dir low and monitors the bus.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: DWC_otg is the driver.</li> <li>■ 1'b1: ULPI PHY is the driver.</li> </ul> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk" :"ulpi_clk,utmi_clk,pmu_clk") : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk" :"ulpi_clk,utmi_clk,utmifs_clk48,pmu_clk,phy_clk")) : "ulpi_clk,phy_clk,pmu_clk") : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ? (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" :"ulpi_clk,utmi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" :"ulpi_clk,utmi_clk,utmifs_clk48,phy_clk")) : (OTG_ENABLE_LPM==1 ? (OTG_FSPHY_INTERFACE==0 ? (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" :"ulpi_clk,utmi_clk,phy_clk") : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" :"ulpi_clk,utmi_clk,utmifs_clk48,phy_clk")) : "ulpi_clk,phy_clk"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-10 ULPI Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
ulpi_nxt	I	<p>Next Data Control.</p> <ul style="list-style-type: none"> <li>■ When the DWC_otg core is sending data to the PHY, this signal indicates when the current byte is accepted by the PHY. The DWC_otg core places the next byte the data bus in the following clock cycle.</li> <li>■ When the PHY is sending data to the DWC_otg core, this signal indicates when a new byte is available for the DWC_otg core.</li> </ul> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)  <b>Synchronous To:</b> ((OTG_EN_PWRROPT==2)    (OTG_EN_PWRROPT==3)    (OTG_AD_P_SUPPORT==1)) ? "ulpi_clk,pmu_clk" : "ulpi_clk"  <b>Registered:</b> No  <b>Power Domain:</b> Vdd  <b>Active State:</b> High</p>

## 5.11 I2C Interface Signals



**Table 5-11 I2C Interface Signals**

Port Name	I/O	Description
i2c_scl	O	<p>I2C Serial Clock. I2C serial clock driven by the DWC_otg core as the I2C Master.</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE != 0 &amp;&amp; (OTG_I2C_INTERFACE == 1    OTG_I2C_INTERFACE == 2))</p> <p><b>Synchronous To:</b> OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,{utmifs_clk48}" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,{utmifs_clk48}" : "utmi_clk,ulpi_clk,{utmifs_clk48}")): "utmifs_clk48"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> N/A</p>
i2c_sda_out	O	<p>Serial Output Data. Inverse I2C serial data driven by the OTG core as the I2C Master.</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE != 0 &amp;&amp; (OTG_I2C_INTERFACE == 1    OTG_I2C_INTERFACE == 2))</p> <p><b>Synchronous To:</b> OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,{utmifs_clk48}" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,{utmifs_clk48}" : "utmi_clk,ulpi_clk,{utmifs_clk48}")): "utmifs_clk48"</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-11 I2C Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
i2c_sda_in	I	<p>Serial Input Data. I2C serial data driven to the DWC_otg core from the I2C device.</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE != 0 &amp;&amp; (OTG_I2C_INTERFACE == 1    OTG_I2C_INTERFACE == 2))</p> <p><b>Synchronous To:</b> OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,{utmifs_clk48}" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,{utmifs_clk48}" : "utmi_clk,ulpi_clk,{utmifs_clk48}")): "utmifs_clk48"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vdd" : "Vdd"}</p> <p><b>Active State:</b> High</p>
i2c_int_n	I	<p>Interrupt Request. Indicates a read request to the DWC_otg core from the I2C device.</p> <p><b>Exists:</b> (OTG_FSPHY_INTERFACE != 0 &amp;&amp; (OTG_I2C_INTERFACE == 1    OTG_I2C_INTERFACE == 2))</p> <p><b>Synchronous To:</b> OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,{utmifs_clk48}" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,{utmifs_clk48}" : "utmi_clk,ulpi_clk,{utmifs_clk48}")): "utmifs_clk48"</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vdd" : "Vdd"}</p> <p><b>Active State:</b> Low</p>

## 5.12 LPM Interface Signals



**Table 5-12 LPM Interface Signals**

Port Name	I/O	Description
utmi_sleep_n	O	<p>UTMI Sleep. The core asserts this active-low signal to indicate that the core is in the Sleep (L1) state, and the PHY can use this signal to enter Shallow Low-Power mode in the Sleep state. In Shallow Low-Power mode, the PHY can switch off some of its internal logic but still provide a PHY clock output by keeping oscillator circuitry alive.</p> <p>When this signal is asserted low, the UTMI PHY transitions to Shallow Low-Power mode by powering down necessary blocks.</p> <p>When this signal is de-asserted (high), the PHY returns to its normal operating state.</p> <p>The core asserts this signal when it transitions to L1 (SLEEP) power state after a successful LPM transaction and is waiting for a Minimum TL1token retry time of 8's and TL1Residency time of 50's.</p> <p>This signal is de-asserted when the power state exits the SLEEP (L1) state, due either to host-initiated resume or device-initiated remote wakeup.</p> <p>The utmi_sleep_n signal is asserted (1'b0) when GLPMCFG.EnbISlpM = 1 and either HIRD_Thres[4] bit = 0 or the received HIRD is too small.</p> <p>When LPM support is not enabled, this signal is always deasserted (driven high).</p> <p><b>Note:</b> This signal is present even when the ULPI interface is configured through coreConsultant.</p> <p><b>Exists:</b> (OTG_ENABLE_LPM==1)</p> <p><b>Synchronous To:</b> OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")): "utmifs_clk48")</p> <p><b>Registered:</b> ((OTG_EN_PWR0OPT==2)    (OTG_EN_PWR0OPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_PWR_CLAMP==0 ? "No" : "Yes") : "Yes"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> Low</p>

**Table 5-12 LPM Interface Signals (Continued)**

Port Name	I/O	Description
utmi_l1_suspend_n	O	<p>UTMI Suspend in L1. This active low signal is asserted from the Core indicating that the core is in Sleep (L1) state and the PHY can use this signal to enter deep low power mode in sleep state. Deep low power mode is wherein the PHY is able to switch off its internal logic as well as the oscillator circuitry. The PHY clock to core will be switched off.</p> <p>When this signal is asserted low, the UTMI PHY transitions to Deep Low-Power mode by powering down its internal logic as well as the oscillator circuitry. If this signal is de-asserted (high), the PHY returns to its normal operating state.</p> <p>The core asserts this signal when it transitions to L1 (SLEEP) power state after a successful LPM transaction and is waiting for a Minimum TL1 token retry time of 50's. This signal is de-asserted when the power state exits the SLEEP (L1) state, whether due to host-initiated resume or device-initiated remote wakeup.</p> <p>The utmi_l1_suspend_n signal is asserted when the received HIRD value carried in the TOKEN is greater than or equal to GLPMCFG.HIRD_Thres[3:0] and the GLPMCFG.HIRD_Thres[4] is set to 1'b1.</p> <p>When LPM support is not enabled, this signal is always deasserted (driven high). This signal is present only if OTG_ENABLE_LPM is enabled.</p> <p><b>Note:</b> This signal is present even when the ULPI interface is configured through coreConsultant.</p> <p><b>Exists:</b> (OTG_ENABLE_LPM==1)</p> <p><b>Synchronous To:</b> OTG_FSPHY_INTERFACE==0  ? (OTG_HSPHY_INTERFACE==1?"utmi_clk" :  (OTG_HSPHY_INTERFACE==2?"ulpi_clk" :"utmi_clk,ulpi_clk")) :  (OTG_HSPHY_INTERFACE!=0?  (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" :  (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" :  "utmi_clk,ulpi_clk,utmifs_clk48")):"utmifs_clk48")</p> <p><b>Registered:</b> ((OTG_EN_PWRROPT==2)    (OTG_EN_PWRROPT==3)     (OTG_AD_P_SUPPORT==1)) ? (OTG_PWR_CLAMP==0 ? "No" :  "Yes") : "Yes"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> Low</p>

**Table 5-12 LPM Interface Signals (Continued)**

Port Name	I/O	Description
dev_hird_vld_tgl	O	<p>This signal is toggled whenever dev_hird_rcvd signal is updated the entry to L1 state after successful L1RetryTimer expiry.</p> <p><b>Configuration:</b> Device LPM only</p> <p><b>Exists:</b> (OTG_ENABLE_LPM==1 &amp;&amp; OTG_MODE!=6 &amp;&amp; OTG_MODE!=5)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADSP_SUPPORT==1)) ? "pmu_clk" : (OTG_EN_PWROPT==0 ? "phy_clk" : (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")): "utmifs_clk48")))</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> Toggle</p>
dev_hird_rcvd[3:0]	O	<p>This signal indicates the latest HIRD received from the host by the device core.</p> <p><b>Exists:</b> (OTG_ENABLE_LPM==1 &amp;&amp; OTG_MODE!=6 &amp;&amp; OTG_MODE!=5)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "phy_clk" : (OTG_PWR_CLAMP==1 ? "phy_clk" : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")): "utmifs_clk48,phy_clk")) : "phy_clk"))</p> <p><b>Registered:</b> ((OTG_EN_PWROPT != 0)    (OTG_ADSP_SUPPORT ==1))&amp;&amp;(OTG_PWR_CLAMP==0)? "No":"Yes"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> N/A</p>

## 5.13 HSIC Interface Signals



**Table 5-13 HSIC Interface Signals**

Port Name	I/O	Description
hsic_if_mode	O	<p>HSIC Interface Mode Active. Indicates that HSIC mode is configured to be active. The signal is set when the OTG_ENABLE_HSIC parameter is selected as 1, if_select_hsic input is driven to 1, and GLPMCFG.InvSelHsic =0. The signals the UTMI interface connected to the PHY will follow HSIC signaling when hsic_if_mode is 1.</p> <p><b>Exists:</b> (OTG_ENABLE_HSIC==1)  <b>Synchronous To:</b> (OTG_EN_PWRLOPT==2    OTG_EN_PWRLOPT==3    OTG_ADP_SUPPORT==1) ? "pmu_hclk": "None"  <b>Registered:</b> No  <b>Power Domain:</b> Vdd  <b>Active State:</b> High</p>
if_select_hsic	I	<p>HSIC Interface Select. Used to select the HSIC mode of operation. Indicates that the HSIC interface is selected. The core starts to connect/operate in HSIC mode when the GLPMCFG.HSICCon is programmed to 1, if GLPMCFG.InvSelHsic = 0.</p> <p><b>Exists:</b> (OTG_ENABLE_HSIC==1)  <b>Synchronous To:</b> None  <b>Registered:</b> No  <b>Power Domain:</b> {OTG_EN_PWRLOPT==1 ? "Vdd" : OTG_EN_PWRLOPT==2 ? "Vsw" : OTG_EN_PWRLOPT==3 ? "Vsw" : "Vdd"}  <b>Active State:</b> High</p>

## 5.14 IC\_USB Interface Signals

ic_usb_dp_in	- ic_usb_dm_out
ic_usb_dm_in	- ic_usb_dp_out
ic_usb_iddig	- ic_usb_dpdः_oe
ic_usb_vbusvalid	- ic_usb_sw1_en
	- ic_usb_sw2_en
	- ic_usb_sw3_en
	- ic_usb_sw4_en
	- ic_usb_sw5_en
	- ic_usb_sw6_en
	- ic_usb_fs_not_ls
	- ic_usb_mode_enable
	- ic_usb_sleep_n
	- ic_usb_suspend_n

**Table 5-14 IC\_USB Interface Signals**

Port Name	I/O	Description
ic_usb_dp_in	I	<p>IC_USB Single-Ended D+ Line. Single-ended D+ signal from the transceiver.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P_SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk,pmu_clk")):(OTG_FSPHY_INTERFACE==0)?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" :"utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" :"utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")):"utmifs_clk48,phy_clk"))</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-14 IC\_USB Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
ic_usb_dm_in	I	<p>IC_USB Single-Ended D' Line. Single-ended D' signal from the transceiver.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWR0PT==2)    (OTG_EN_PWR0PT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk,pmu_clk" : "utmi_clk,ulpi_clk,phy_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk,pmu_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk"))</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
ic_usb_dm_out	O	<p>Single-ended D' Line. Single-ended D' line driver. It is valid only when ic_usb_dpdp_oe is asserted (high).</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWR0PT==2)    (OTG_EN_PWR0PT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk" : "utmi_clk,ulpi_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk")) : "utmifs_clk48,pmu_clk")) : (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48"))</p> <p><b>Registered:</b> ((OTG_EN_PWR0PT == 2)    (OTG_EN_PWR0PT == 3)    (OTG_AD_P SUPPORT ==1)) ? "No":"Yes"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-14 IC\_USB Interface Signals (Continued)**

Port Name	I/O	Description
ic_usb_dp_out	O	<p>Single-ended D+ Line. Single-ended D+ line driver. It is valid only when ic_usb_dpdः_oe is asserted (high).</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P_SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk" : "utmi_clk,ulpi_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk")) : "utmifs_clk48,pmu_clk")) : (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48"))</p> <p><b>Registered:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT ==1))? "No": "Yes"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
ic_usb_dpdः_oe	O	<p>Output Enable. Data valid and output enable for ic_usb_dp_out and ic_usb_dm_out.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P_SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk,phy_clk" : "utmi_clk,ulpi_clk,pmu_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk,phy_clk")) : "utmifs_clk48,pmu_clk,phy_clk")) : (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-14 IC\_USB Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
ic_usb_sw1_en	O	<p>SW1 Enable for Host. When asserted, signals the OTG transceiver to close SW1 connected to pull-down resistor RPDH the D+ line, as shown in the USB Engineering Change Notice (ECN): Inter-Chip USB Supplement to the USB 2.0 Specification. This signal controls SW1 in the ECN.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk" : "utmi_clk,ulpi_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk")) : "utmifs_clk48,pmu_clk")) : (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> HIGH</p>

**Table 5-14 IC\_USB Interface Signals (Continued)**

Port Name	I/O	Description
ic_usb_sw2_en	O	<p>SW2 Enable for Host. When asserted, signals the OTG transceiver to close SW1 connected to pull-down resistor RPDH the D' line, as shown in the USB Engineering Change Notice (ECN): Inter-Chip USB Supplement to the USB 2.0 Specification. This signal controls SW2 in the ECN.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk")) : (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> HIGH</p>

**Table 5-14 IC\_USB Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
ic_usb_sw3_en	O	<p>SW3 Enable for Device. When asserted, signals the OTG transceiver to close SW3 connected to pull-down resistor RPDP the D+ line, as shown in the USB Engineering Change Notice (ECN): Inter-Chip USB Supplement to the USB 2.0 Specification. This signal controls SW3 in the ECN.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) : (OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk,phy_clk" : "utmi_clk,ulpi_clk,pmu_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk,phy_clk" : "utmifs_clk48,pmu_clk,phy_clk")) : (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> HIGH</p>

**Table 5-14 IC\_USB Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
ic_usb_sw4_en	O	<p>SW4 Enable for Device. When asserted, signals the OTG transceiver to close SW4 connected to pull-down resistor RPDP the D' line, as shown in the USB Engineering Change Notice (ECN): Inter-Chip USB Supplement to the USB 2.0 Specification. This signal controls SW4 in the ECN.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk,phy_clk" : "utmi_clk,ulpi_clk,pmu_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk,phy_clk" : "utmifs_clk48,pmu_clk,phy_clk")) : (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> HIGH</p>

**Table 5-14 IC\_USB Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
ic_usb_sw5_en	O	<p>SW5 Enable for Device. When asserted, signals the OTG transceiver to close SW5 connected to pull-up resistor RPU2 the D+/D' line, as shown in the USB Engineering Change Notice (ECN): Inter-Chip USB Supplement to the USB 2.0 Specification. This signal controls SW5 in the ECN.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk,phy_clk" : "utmi_clk,ulpi_clk,pmu_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk,phy_clk" : "utmifs_clk48,pmu_clk,phy_clk")) : (OTG_MODE==6 &amp;&amp; OTG_EN_PWROPT==0) ? "None" : (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> HIGH</p>

**Table 5-14 IC\_USB Interface Signals (Continued)**

Port Name	I/O	Description
ic_usb_sw6_en	O	<p>SW6 Enable for Device. When asserted, signals the OTG transceiver to close SW6 connected to pull-up resistor RPU1 the D+/D' line, as shown in the USB Engineering Change Notice (ECN): Inter-Chip USB Supplement to the USB 2.0 Specification. This signal controls SW6 in the ECN.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk,phy_clk" : "utmi_clk,ulpi_clk,pmu_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk,phy_clk" : "utmifs_clk48,pmu_clk,phy_clk")) : (OTG_MODE==6 &amp;&amp; OTG_EN_PWROPT==0) ? "None" : (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48")))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> HIGH</p>
ic_usb_iddig	I	<p>Similar to ID pin for OTG, specifies whether the core is a host or device.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48")) : "phy_clk"</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vdd" : "Vdd"}</p> <p><b>Active State:</b> MISSING-FIELD-ERROR</p>

**Table 5-14 IC\_USB Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
ic_usb_fs_not_ls	O	<p>Indicates the configured speed of the device. High when in Full Speed mode; Low when in Low Speed mode.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk,phy_clk" : "utmi_clk,ulpi_clk,pmu_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk,phy_clk")) : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk")) : "phy_clk")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> HIGH</p>
ic_usb_mode_enable	O	<p>Indicates that IC USB Mode has been enabled and in turn UTMI/FSLS modes have been disabled.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk" : "utmi_clk,ulpi_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk")) : "utmifs_clk48,pmu_clk")) : "None"</p> <p><b>Registered:</b> (OTG_EN_PWROPT==1) ? "Yes" : "No"</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> HIGH</p>

**Table 5-14 IC\_USB Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
ic_usb_sleep_n	O	<p>Indicates that the core is in SLEEP (L1) state as per LPM specifications.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1 &amp;&amp; OTG_ENABLE_LPM==1)</p> <p><b>Synchronous To:</b> OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")): "utmifs_clk48") : "phy_clk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> N/A</p>
ic_usb_suspend_n	O	<p>Indicates that the core is in SUSPEND state.</p> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADP_SUPPORT==1) ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk" : "utmi_clk,ulpi_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk")): "utmifs_clk48,pmu_clk") :(OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")): "utmifs_clk48")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> N/A</p>

**Table 5-14 IC\_USB Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
ic_usb_vbusvalid	I	<p>IC_VDD Valid. Indicates if the voltage VBUS is valid for device/peripheral operation. The comparator thresholds are:</p> <ul style="list-style-type: none"> <li>■ 1'b0: IC_VDD &lt; 0.8 * IC_VDDmin</li> <li>■ 1'b1: IC_VDD &gt; 0.8 * IC_VDDmin IC_VDDmin is defined in the fourth column, second row of Table 5-1 through Table 5-5 of the USB Engineering Change Notice (ECN): Inter-Chip USB Supplement to the USB 2.0 Specification.</li> </ul> <p><b>Exists:</b> (OTG_ENABLE_IC_USB==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADP_SUPPORT==1)) ? (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,pmu_clk" : "utmi_clk,ulpi_clk,pmu_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,pmu_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,pmu_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,pmu_clk")) : "utmifs_clk48,pmu_clk") : ((OTG_MODE!=5 &amp;&amp; OTG_MODE!=6) ? (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,phy_clk" : "utmi_clk,ulpi_clk,phy_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48,phy_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48,phy_clk" : "utmi_clk,ulpi_clk,utmifs_clk48,phy_clk")) : "utmifs_clk48,phy_clk") : (OTG_FSPHY_INTERFACE==0) ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48"))</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

## 5.15 Simulation Control Interface Signals

ss\_scaledown\_mode -

**Table 5-15 Simulation Control Interface Signals**

Port Name	I/O	Description
ss_scaledown_mode[1:0]	I	<p>Scale-Down Mode.</p> <ul style="list-style-type: none"> <li>■ When this signal is enabled during simulation, the core uses scaled-down timing values, resulting in faster simulations.</li> <li>■ When it is disabled, actual timing values are used.</li> </ul> <p><b>Note:</b></p> <p>If you need faster post simulation mode, then tie this signal to 2'b00 without optimizing the related logic in synthesis. Use two programmable registers whose default value is 2'b00 to drive this signal. If you do not require faster post simulation mode, simply tie this signal to 2'b00.</p> <p>This strap signal is tied to one static value.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 2'b00: Disables all scale-downs. Actual timing values are used. Required for synthesis.</li> <li>■ 2'b01: Enables scale-down of all timing values except Device mode suspend and resume. These include: <ul style="list-style-type: none"> <li>- Speed enumeration</li> <li>- HNP/SRP</li> <li>- Host mode suspend and resume</li> </ul> </li> <li>■ 2'b10: Enables scale-down of Device mode suspend and resume timing values only.</li> <li>■ 2'b11: Enables bit 0 and bit 1 scale-down timing values.</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> N/A</p>

## 5.16 Miscellaneous Interface Signals

gp_in	- interrupt
scan_mode	- endp_interrupt
sof_update_toggle	- endp_multi_proc_interrupt
sof_count	- internal_probes
test_bypass_lp	- internal_probes_p
pwr_switch	- soft_reset_actv_pdomain
dbncc_filtr_bypass	- gp_out - sof_toggle_out - sof_sent_rcvd_tgl - internal_probes_cc

Table 5-16 Miscellaneous Interface Signals

Port Name	I/O	Description
interrupt	O	<p>Interrupt Line. Interrupts the application to indicate the occurrence of an event needing application intervention.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Event interrupt disabled</li> <li>■ 1'b1: Event interrupt enabled</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT ==1)) ? "hclk,pmu_hclk":"hclk"</p> <p><b>Registered:</b> ((OTG_EN_PWROPT == 2)    (OTG_EN_PWROPT == 3)    (OTG_AD_P_SUPPORT ==1)) ? "No":"Yes"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-16 Miscellaneous Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
endp_interrupt[31:0]	O	<p>Endpoint-Specific Interrupt Line. Endpoint-specific interrupt to the application, indicating the occurrence of an event a USB endpoint needing application intervention.</p> <p><b>Dependency:</b></p> <p>This is an optional feature and can be selected when the parameter OTG_MULTI_PROC_INTRPT=1. When this option is selected, the interrupts related specific to USB endpoints receive these lines, provided that the corresponding endpoint-specific mask registers are selected.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Event interrupt disabled</li> <li>■ 1'b1: Event interrupt enabled</li> </ul> <p><b>Exists:</b> OTG_MULTI_PROC_INTRPT == 1</p> <p><b>Synchronous To:</b> hclk_gated</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vsw" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
endp_multi_proc_interrupt	O	<p>Single Endpoint-Specific Interrupt Line. Endpoint-specific interrupt to the application, indicating the occurrence of an event a USB endpoint needing application intervention. This is optional feature can be selected when the configuration parameter OTG_MULTI_PROC_INTRPT = 1.</p> <p>This pin is derived by OR-ing the endp_interrupt[31:0] bits with the interrupt pin:</p> <p>endp_multi_proc_interrupt = lendp_interrupt[31:0]   interrupt;</p> <p><b>Dependency:</b> When this option is selected, all interrupts will receive these lines, if the corresponding endpoint-specific mask registers are selected.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Event interrupt disabled</li> <li>■ 1'b1: Event interrupt enabled</li> </ul> <p><b>Exists:</b> OTG_MULTI_PROC_INTRPT == 1</p> <p><b>Synchronous To:</b> hclk</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-16 Miscellaneous Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
internal_probes[61:0]	O	<p>Internal Probe AHB Domain. Used by the verification environment.</p> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> hclk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
internal_probes_p[218:0]	O	<p>Internal Probe PHY Domain. Used by the verification environment.</p> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> phy_clk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
soft_reset_actv_pdomain	O	<p>Indicates that soft reset is active.</p> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")): "utmifs_clk48")) : "phy_clk"</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
gp_out[15:0]	O	<p>General Purpose Output Port. Can be used as general purpose outputs.</p> <p><b>Exists:</b> OTG_RM_OPT_FEATURES != 1</p> <p><b>Synchronous To:</b> hclk</p> <p><b>Registered:</b> (((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADP_SUPPORT==1)) &amp;&amp; (OTG_PWR_CLAMP == 0)) ? "No": "Yes"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-16 Miscellaneous Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
gp_in[15:0]	I	<p>General Purpose Input Port. Can be used as general purpose inputs.</p> <p><b>Exists:</b> OTG_RM_OPT_FEATURES != 1</p> <p><b>Synchronous To:</b> hclk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
scan_mode	I	<p>Scan Mode Port. Enables Scan mode bypass to improve scan coverage.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: Scan mode bypass disabled</li> <li>■ 1'b1: Scan mode bypass enabled</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
sof_update_toggle	I	<p>Update SOF. Toggle input to update SOF value Host Frame Number/Frame Time Remaining Register (HFNUM) Field FrNum.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ Wireless USB Device Wire Adapter (DWA) application: Can be used to synchronize the wired Host SOF to wireless USB time.</li> <li>■ Non-wireless USB application: Must be tied to 0. This signal is not present when parameter OTG_RM_OPT_FEATURES=Yes.</li> </ul> <p><b>Exists:</b> OTG_RM_OPT_FEATURES != 1</p> <p><b>Synchronous To:</b> hclk_gated</p> <p><b>Registered:</b> (OTG_SYNC_RESET_TYPE == 1) ? "No" : "Yes"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==0 ? "Vdd" : "Vsw"}</p> <p><b>Active State:</b> High</p>

**Table 5-16 Miscellaneous Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
sof_count[13:0]	I	<p>SOF Input Count. Wireless USB Wire Adapter (DWA) application</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ Wireless USB Device Wire Adapter (DWA) application: Input value to be loaded into Host Frame Number/Frame Time Remaining Register (HFNUM) field FrNum.</li> <li>■ Non-Wireless USB application: Must be tied to 0.</li> </ul> <p><b>Exists:</b> OTG_RM_OPT_FEATURES != 1  <b>Synchronous To:</b> None  <b>Registered:</b> No  <b>Power Domain:</b> {OTG_EN_PWROPT==0 ? "Vdd" : "Vsw"}  <b>Active State:</b> High</p>
sof_toggle_out	O	<p>In Host mode, this signal toggles every time an SOF is generated. In Device mode, this signal toggles every time an SOF token is received from the USB host or when an SOF token is missed at the start of frame.</p> <p><b>Exists:</b> Always  <b>Synchronous To:</b> phy_clk  <b>Registered:</b> Yes  <b>Power Domain:</b> {OTG_EN_PWROPT==0 ? "Vdd" : "Vsw"}  <b>Active State:</b> N/A</p>
sof_sent_rcvd_tgl	O	<p>In Host mode, this signal toggles every time an SOF is successfully transmitted. In Device mode, this signal toggles only when a valid SOF token is received from the USB host.</p> <p><b>Configuration:</b> All configurations  <b>Exists:</b> Always  <b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P_SUPPORT==1)) ? "pmu_clk" : (OTG_EN_PWROPT!=0 ? (OTG_FSPHY_INTERFACE==0 ?(OTG_HSPHY_INTERFACE==1?"utmi_clk" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk" : "utmi_clk,ulpi_clk")) : (OTG_HSPHY_INTERFACE!=0? (OTG_HSPHY_INTERFACE==1?"utmi_clk,utmifs_clk48" : (OTG_HSPHY_INTERFACE==2?"ulpi_clk,utmifs_clk48" : "utmi_clk,ulpi_clk,utmifs_clk48")) : "utmifs_clk48")) : "phy_clk")  <b>Registered:</b> Yes  <b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}  <b>Active State:</b> Toggle</p>

**Table 5-16 Miscellaneous Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
test_bypass_lp	I	<p>This port is used to bypass Isolation cells inserted by Synthesis Tools using the Unified Power Format flow during DFT</p> <p><b>Exists:</b> (OTG_EN_PWROPT!=0)</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
pwr_switch	I	<p>This port is required to connect the Power Switch Control pin defined in the Unified Power Format for configurations which support the Partial Power Down Feature.</p> <p><b>Exists:</b> (OTG_EN_PWROPT == 1)</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> Low</p>
dbnce_filtr_bypass	I	<p>Bypass Debounce filters for avalid, bvalid, vbusvalid, sessend, iddig signals when enabled.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 1'b1: Enabled</li> <li>■ 1'b0: Disabled</li> </ul> <p><b>Exists:</b> ((OTG_EN_IDDIG_FILTER == 1)    (OTG_EN_VBUSVALID_FILTER == 1)    (OTG_EN_A_VALID_FILTER == 1)    (OTG_EN_B_VALID_FILTER == 1)    (OTG_EN_SESSIONEND_FILTER == 1))</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vsw" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-16 Miscellaneous Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
internal_probes_cc[56:0]	O	<p>Internal Probe Type-C cc_clk Domain. Used by the verification environment.</p> <p><b>Exists:</b> OTG_TYPEC_MODE==1</p> <p><b>Synchronous To:</b> (OTG_MODE==0    OTG_MODE==1    OTG_MODE==2) ? "0:6=cc_clk;7:9=None;10:22=None;23:23=cc_clk;24:28=cc_clk;29:30=None;31:35=cc_clk;36:37=None;38:38=cc_clk;39:46=cc_clk;47:49=cc_clk;50:51=cc_clk;52:55=None;56:56=cc_clk" : ((OTG_MODE==3    OTG_MODE==4) ? "0:6=cc_clk;7:9=None;10:22=None;23:23=None;24:28=cc_clk;29:30=None;31:35=cc_clk;36:37=None;38:38=cc_clk;39:42=cc_clk;43:46=None;47:49=cc_clk;50:51=None;52:55=None;56:56=cc_clk" : "0:6=cc_clk;7:9=None;10:22=None;23:23=None;24:28=cc_clk;29:30=None;31:35=cc_clk;36:37=None;38:38=cc_clk;39:42=None;43:46=cc_clk;47:49=cc_clk;50:51=None;52:55=None;56:56=cc_clk")</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

## 5.17 Remote Memory Support Interface Signals



**Table 5-17 Remote Memory Support Interface Signals**

Port Name	I/O	Description
sys_dma_done	I	<p>System DMA Done. This signal should be asserted when the DATA write is completed in the System Memory. It should be asserted for one AHB clock cycle synchronous to hclk. The signal is valid only when RMS is enabled.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0: Data write not complete.</li> <li>■ 1: Data Write complete in the system memory for the current DMA write-transfer from the controller</li> </ul> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)  <b>Synchronous To:</b> hclk_gated  <b>Registered:</b> No  <b>Power Domain:</b> {OTG_EN_PWROPT==0 ? "Vdd" : "Vsw"}  <b>Active State:</b> High</p>
int_dma_req	O	<p>Internal DMA Request. This signal is the DMA request for DATA write from the controller. It is asserted until the sys_dma_done is asserted from the system. The signal is valid only when RMS is enabled.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0: No DMA transfer</li> <li>■ 1: Active DMA write transfer from the AHB master.</li> </ul> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)  <b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "hclk,hclk_gated" : "hclk_gated") : "hclk_gated"  <b>Registered:</b> No  <b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}  <b>Active State:</b> High</p>

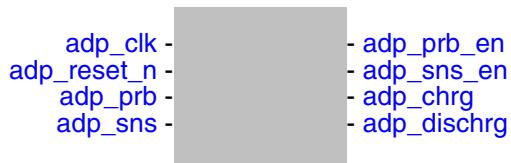
**Table 5-17 Remote Memory Support Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
int_dma_done	O	<p>Internal DMA Done. This signal is the Internal DMA done from the controller. It is asserted for one AHB clock whenever DMA transfer is done at the controller boundary. The signal is valid only when RMS is enabled.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0: DMA transfer not done</li> <li>■ 1: Current DMA transfer from the controller is done</li> </ul> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "hclk,hclk_gated" : "hclk_gated") : "hclk_gated"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
chep_number[3:0]	O	<p>Channel/Endpoint Number. The Channel/Endpoint number info for the ongoing DMA WRITE transfer. The signal is valid only when RMS is enabled.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0000: Channel/Endpoint 0</li> <li>■ 0001: Channel/Endpoint 1</li> <li>■ 0010: Channel/Endpoint 2</li> <li>■ and so on up to,</li> <li>■ 1111: Channel/Endpoint 15</li> </ul> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "hclk,hclk_gated" : "hclk_gated") : "hclk_gated"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> NA</p>

**Table 5-17 Remote Memory Support Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
chep_last_trans	O	<p>Channel/Endpoint Last Transaction. The signal is asserted for the last transaction of the DMA write corresponding to the Channel/Endpoint. The signal is valid only when RMS is enabled.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0: Current DMA write is not the last transaction</li> <li>■ 1: Current DMA write is the last transaction</li> </ul> <p><b>Exists:</b> (OTG_ARCHITECTURE == 2)</p> <p><b>Synchronous To:</b> (OTG_EN_PWROPT==2    OTG_EN_PWROPT==3) ? "hclk_gated" : OTG_EN_PWROPT!=0 ? (OTG_PWR_CLAMP==0 ? "hclk,hclk_gated" : "hclk_gated") : "hclk_gated"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

## 5.18 ADP Interface Signals



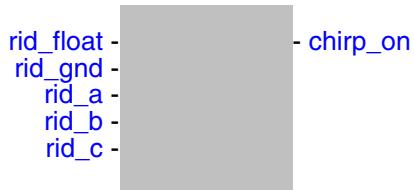
**Table 5-18 ADP Interface Signals**

Port Name	I/O	Description
adp_clk	I	<p>32-kHz frequency clock for ADP operations.</p> <p><b>Value:</b> N/A</p> <p><b>Exists:</b> (OTG_ADP_SUPPORT==1)</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> N/A</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> N/A</p>
adp_reset_n	I	<p>Reset signal for logic running adp_clk.</p> <p><b>Value:</b> N/A</p> <p><b>Exists:</b> (OTG_ADP_SUPPORT==1)</p> <p><b>Synchronous To:</b> None</p> <p><b>Registered:</b> N/A</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> Low</p>
adp_prb_en	O	<p>ADP probe enable indication to PHY.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_ADP_SUPPORT==1)</p> <p><b>Synchronous To:</b> adp_clk</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>

**Table 5-18 ADP Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
adp_sns_en	O	<p>ADP Sense enable indication to PHY.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_ADP_SUPPORT==1)</p> <p><b>Synchronous To:</b> adp_clk</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
adp_chrg	O	<p>ADP Charge indication to PHY to charge VBUS.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_ADP_SUPPORT==1)</p> <p><b>Synchronous To:</b> adp_clk</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
adp_dischrg	O	<p>ADP discharge indication to PHY to discharge VBUS.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_ADP_SUPPORT==1)</p> <p><b>Synchronous To:</b> adp_clk</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>
adp_prb	I	<p>Probe threshold Voltage hit indication by the PHY.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_ADP_SUPPORT==1)</p> <p><b>Synchronous To:</b> adp_clk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
adp_sns	I	<p>Sense threshold Voltage hit indication by the PHY.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_ADP_SUPPORT==1)</p> <p><b>Synchronous To:</b> adp_clk</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

## 5.19 Battery Charger ACA Interface Signals



**Table 5-19 Battery Charger ACA Interface Signals**

Port Name	I/O	Description
rid_float	I	<p>Measured RID_A resistance of the IDDIG pin.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_BC_SUPPORT==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? "hclk,pmu_hclk":"hclk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
rid_gnd	I	<p>Measured RID_GND resistance of the IDDIG pin. If this value is 1, it indicates a A-device mode of operation.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_BC_SUPPORT==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? "hclk,pmu_hclk":"hclk"</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
rid_a	I	<p>Measured RID_A resistance of the IDDIG pin. If this value is 1, it indicates a A-device mode of operation.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_BC_SUPPORT==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_AD_P SUPPORT==1)) ? "hclk,pmu_hclk":"hclk"</p> <p><b>Registered:</b> Yes</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>

**Table 5-19 Battery Charger ACA Interface Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
rid_b	I	<p>Measured RID_B resistance of the IDDIG pin.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_BC_SUPPORT==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADAP_SUPPORT==1)) ? "hclk,pmu_hclk":"hclk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
rid_c	I	<p>Measured RID_C resistance of the IDDIG pin.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_BC_SUPPORT==1)</p> <p><b>Synchronous To:</b> ((OTG_EN_PWROPT==2)    (OTG_EN_PWROPT==3)    (OTG_ADAP_SUPPORT==1)) ? "hclk,pmu_hclk":"hclk"</p> <p><b>Registered:</b> No</p> <p><b>Power Domain:</b> Vdd</p> <p><b>Active State:</b> High</p>
chirp_on	O	<p>When asserted indicates an imminent chirp signal.</p> <p><b>Values:</b> 0,1</p> <p><b>Exists:</b> (OTG_BC_SUPPORT==1)</p> <p><b>Synchronous To:</b> phy_clk</p> <p><b>Registered:</b> {OTG_PWR_CLAMP} == 1 ? "Yes":"No"</p> <p><b>Power Domain:</b> {OTG_EN_PWROPT==1 ? "Vdd" : OTG_EN_PWROPT==2 ? "Vdd" : OTG_EN_PWROPT==3 ? "Vsw" : "Vdd"}</p> <p><b>Active State:</b> High</p>



# 6

## Control and Status Registers

---



**Note** For description of registers mentioned in this chapter, see [Chapter 7, “Register Descriptions”](#).

---

This chapter provides the memory map and descriptions for the DWC\_otg controller’s Control and Status registers (CSRs).

### 6.1 Control and Status Registers Overview

Your application controls the DWC\_otg controller by reading from and writing to the Control and Status Registers (CSRs) through the AHB Slave interface. These registers are 32 bits wide and the addresses are 32-bit block aligned.

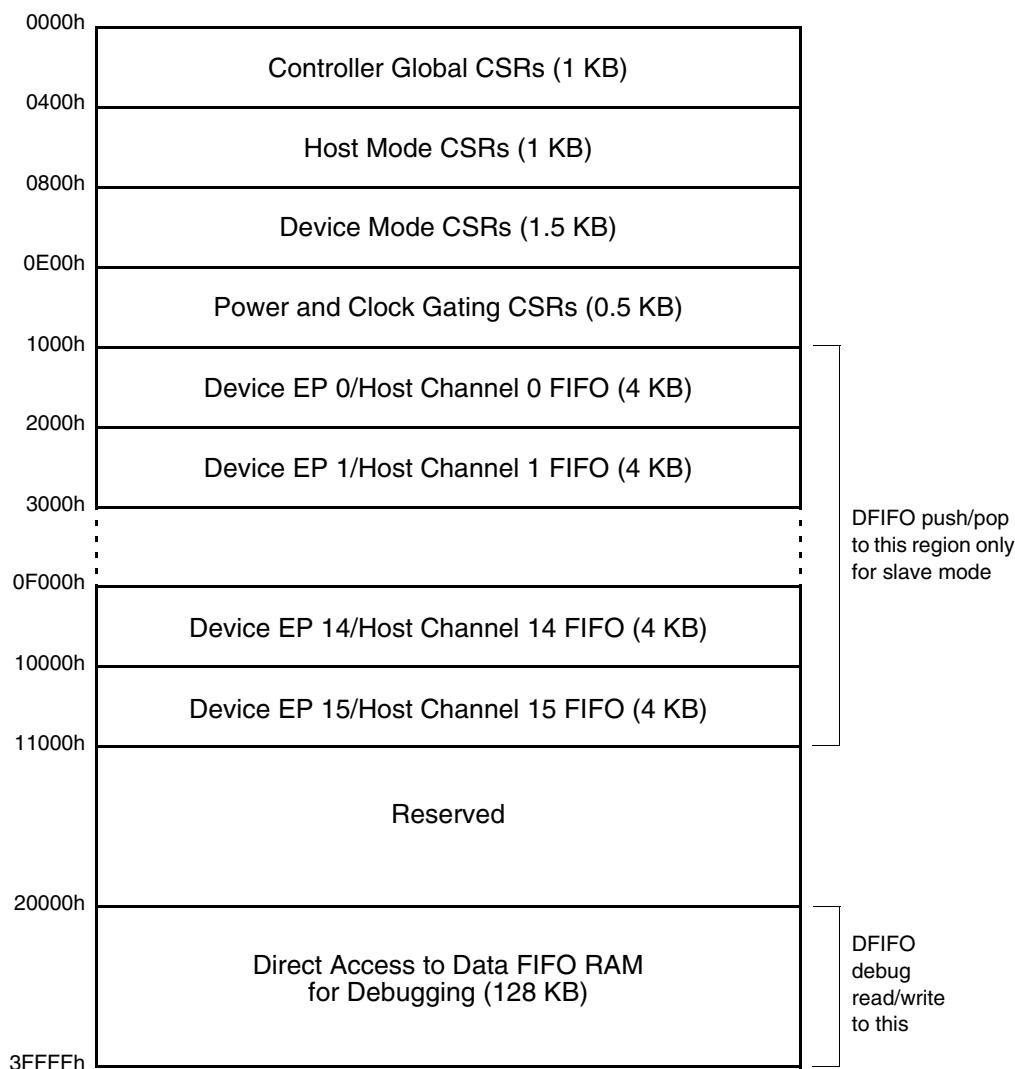
Only the Controller Global, Power and Clock Gating, Data FIFO Access, and Host Port registers can be accessed in both Host and Device modes. When the DWC\_otg controller is operating in one mode, either Device or Host, the application must not access registers from the other mode. If an illegal access occurs, a Mode Mismatch interrupt is generated and reflected in the Controller Interrupt register (GINTSTS.ModeMis).

When the controller switches from one mode to another, the registers in the new mode must be reprogrammed as they would be after a power-on reset.

### 6.2 CSR Memory Map

The CSR address map is fixed and does not depend on the controller’s configuration (for example, how many endpoints are implemented). Host and Device mode registers occupy different addresses. All registers are implemented in the AHB Clock domain. See [“AHB Bus Interface Unit \(BIU\)”](#) on page [71](#) for FIFO implementation and mapping details.

[Figure 6-1](#) shows the CSR address map.

**Figure 6-1 OTG CSR Memory Map**

The tables in this section provide high-level summaries of each register and register group.

Register Name	Name of register types and register names ordered by offset address
Acronym	Shorthand names for registers that are mapped to the offset address. These are used extensively in the programming examples in the Programming Guide. The first letter is a prefix for the register type: G: Controller Global H: Host mode D: Device mode
Offset Address	Address, in hexadecimal (h), of the first byte of each register.



FIFO size and FIFO depth are used interchangeably.

## Global CSRs

These registers are available in both Host and Device modes.

### Host Mode CSR Map

These registers must be programmed every time the controller changes to Host mode.

### Device Mode CSR Map

These registers must be programmed every time the controller changes to Device mode.

### Data FIFO (DFIFO) Access Register Map

These registers, available in both Host and Device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

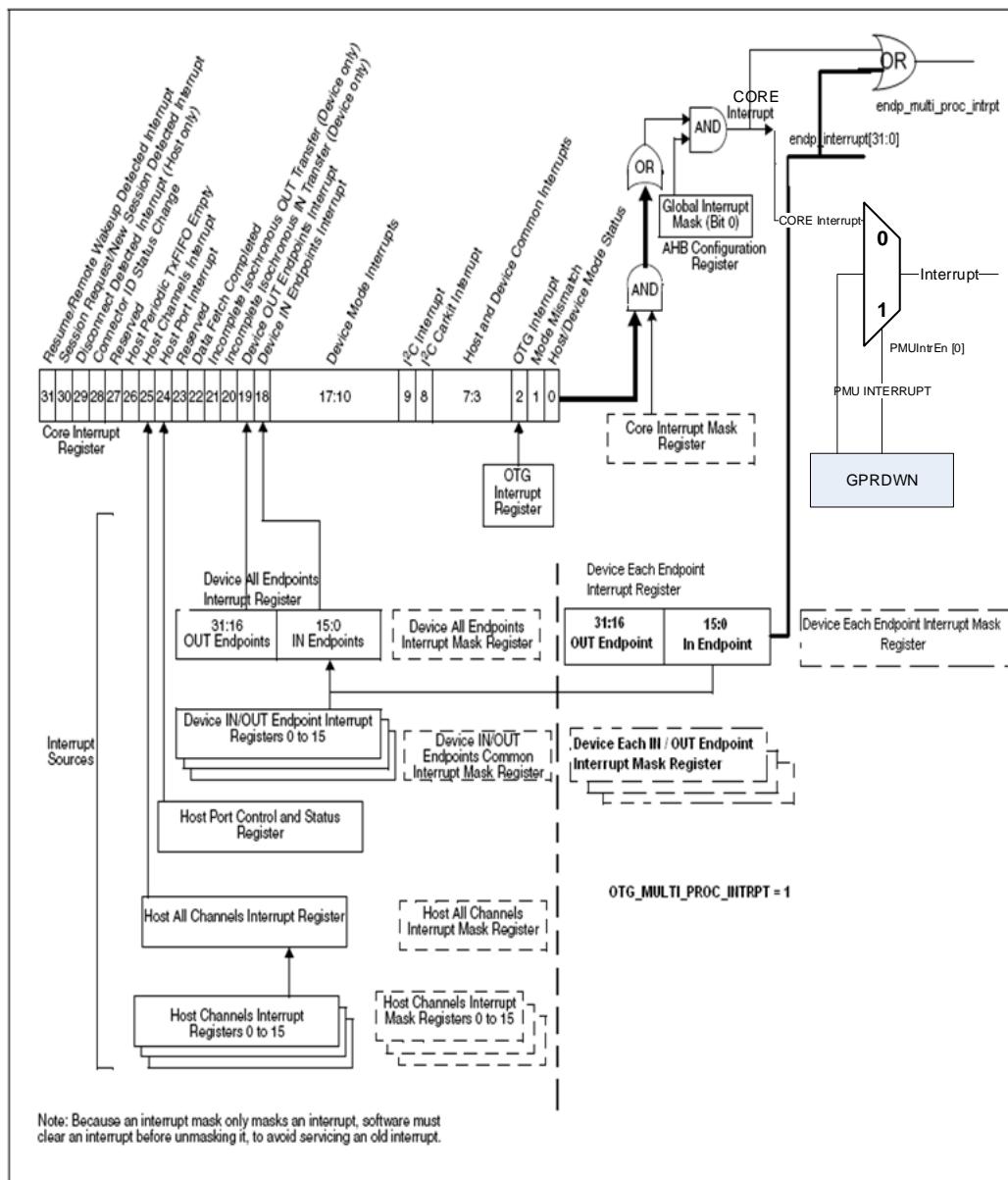
### Power and Clock Gating CSR Map

There is a single register for power and clock gating. It is available in both Host and Device modes.

## 6.2.1 Interrupt Hierarchy

Figure 6-2 displays the DWC\_otg interrupt hierarchy.

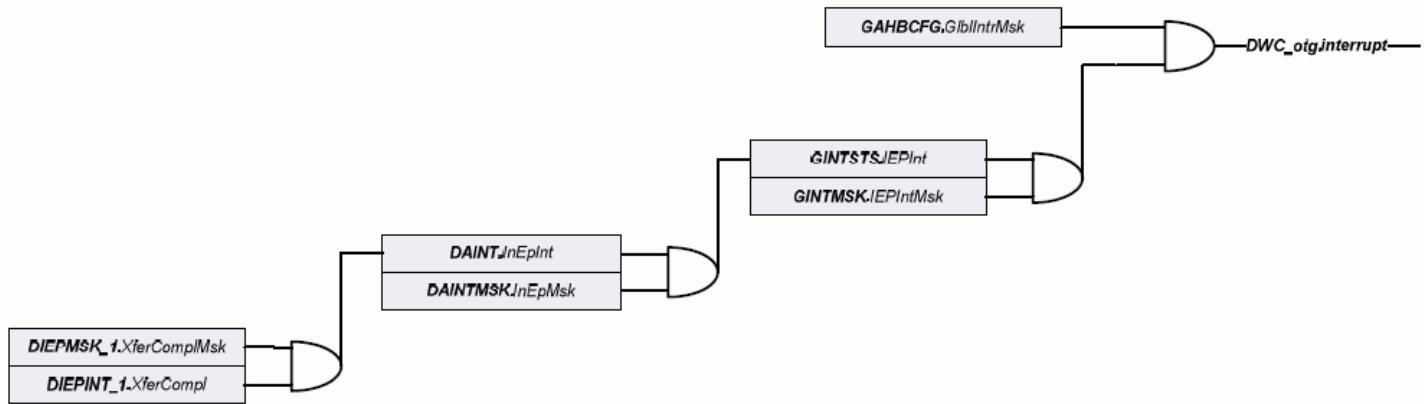
## **Figure 6-2    Interrupt Hierarchy**



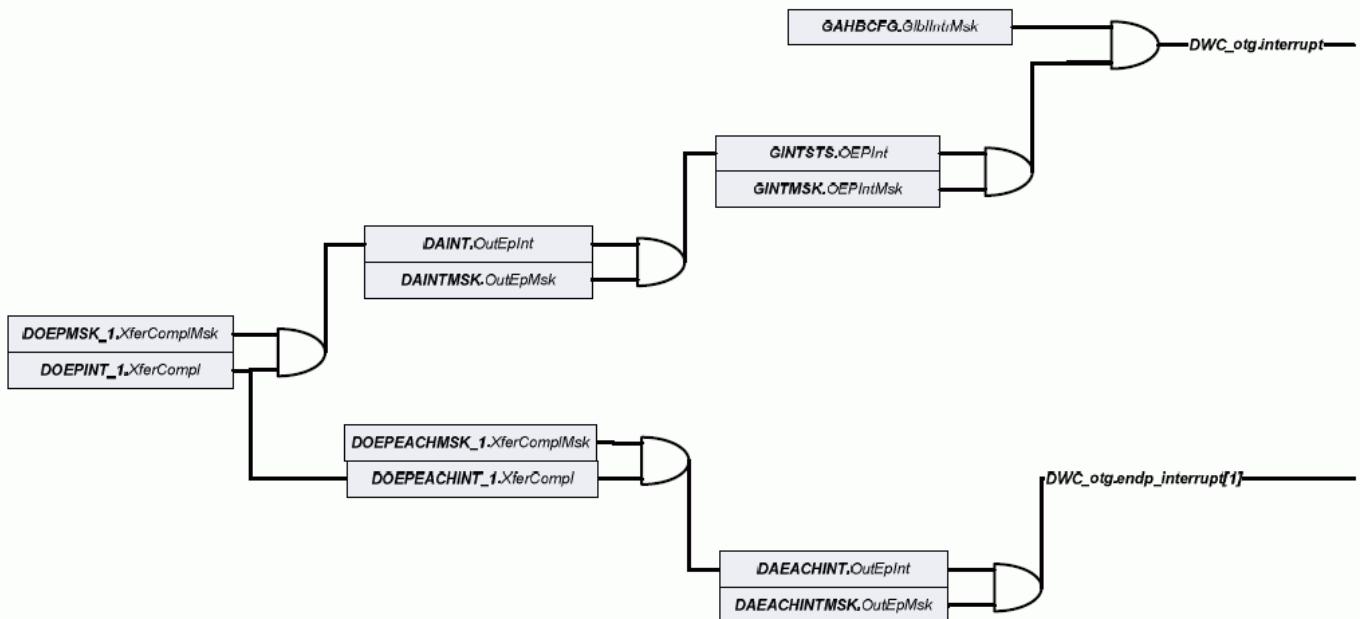
### 6.2.1.1 Multi Processor Interrupts

Figures 6-3 and 6-4 illustrate the interrupt structure when MULTI\_PROC\_INTRPT is set to 0 or 1.

**Figure 6-3** Interrupt Structure when MULTI\_PROC\_INTRPT=0



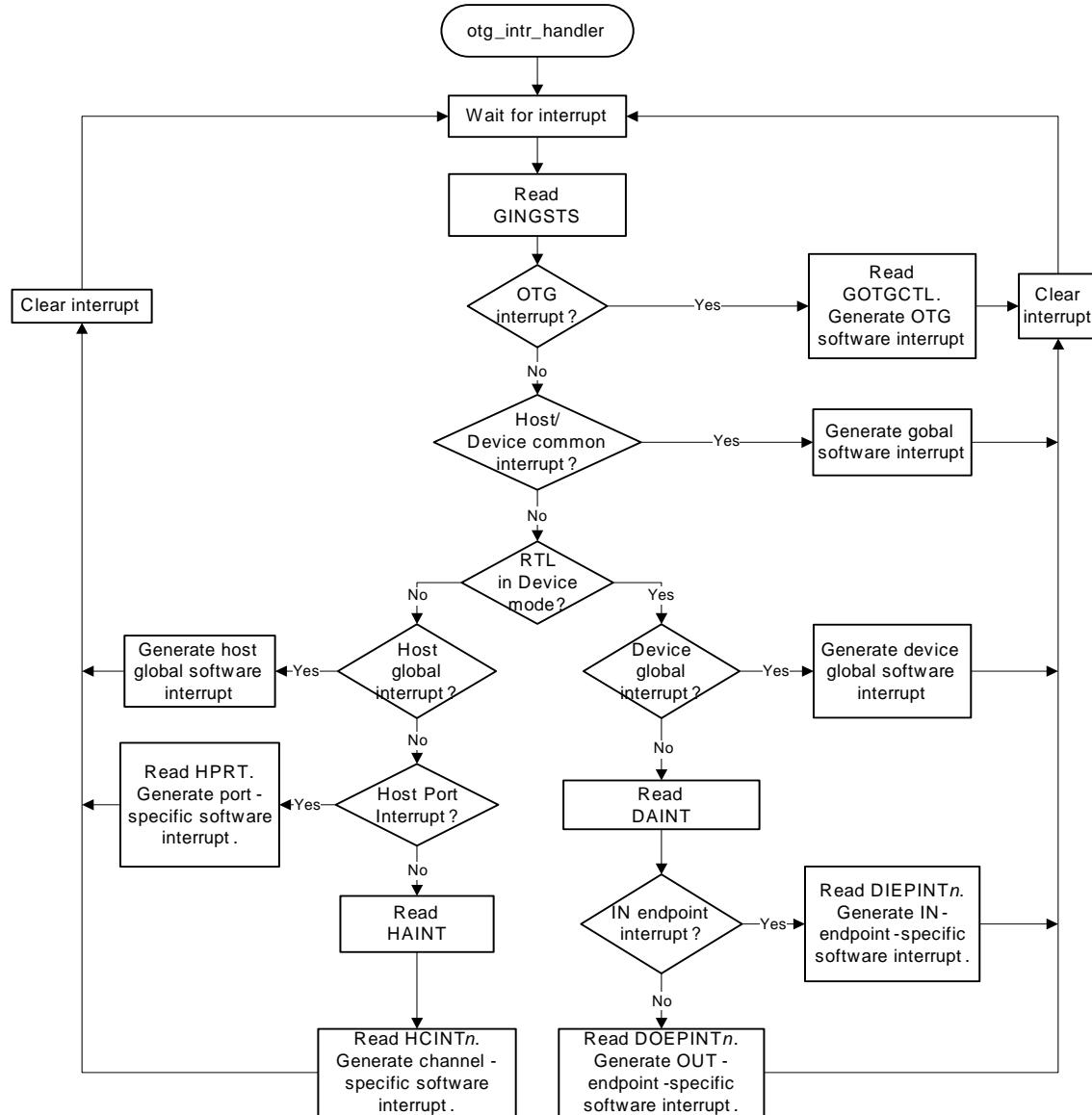
**Figure 6-4** Interrupt Structure when MULTI\_PROC\_INTRPT=1



### 6.2.1.2 Controller Interrupt Handler

Figure 6-5 illustrates how the controller interrupt handler works.

**Figure 6-5 Controller Interrupt Handler**



## 6.3 Overview of Commonly Used Register Bits

This section provides an overview of the commonly used registers and bits. For a complete description of all the registers, see the corresponding sections.

**Table 6-1 List of Commonly Used Register Bits**

Bit Number	Register/Bit Name	Description
	<b>Control and Status Register (GOTGCTL)</b>	The OTG Control and Status register controls the behavior and reflects the status of the OTG function of the controller.
11	<b>Device HNP Enabled (DevHNPEn)</b>	The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host. <ul style="list-style-type: none"> <li>■ 1'b0: HNP is not enabled in the application</li> <li>■ 1'b1: HNP is enabled in the application</li> </ul>
10	<b>Host Set HNP Enable (HstSetHNPEn)</b>	The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device. <ul style="list-style-type: none"> <li>■ 1'b0: Host Set HNP is not enabled</li> <li>■ 1'b1: Host Set HNP is enabled</li> </ul>
	<b>AHB Configuration Register (GAHBCFG)</b>	This register can be used to configure the controller after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.
5	<b>DMA Enable (DMAEn)</b>	This bit is always 0 when Slave-Only mode has been selected for the Architecture in coreConsultant (parameter OTG_ARCHITECTURE = 0). <ul style="list-style-type: none"> <li>■ 1'b0: Controller operates in Slave mode</li> <li>■ 1'b1: Controller operates in a DMA mode</li> </ul>
	<b>USB Configuration Register (GUSBCFG)</b>	This register can be used to configure the controller after power-on or a changing to Host mode or Device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.
30	<b>Force Device Mode (ForceDevMode)</b>	Writing a 1 to this bit forces the controller to device mode irrespective of utmiotg_iddig input pin. <ul style="list-style-type: none"> <li>■ 1'b0: Normal Mode</li> <li>■ 1'b1: Force Device Mode</li> </ul> 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 $\mu$ s is sufficient. This bit is valid only when OTG_MODE = 0, 1 or 2. In all other cases, this bit reads 0.

Bit Number	Register/Bit Name	Description
29	<b>Force Host Mode (ForceHstMode)</b>	<p>Writing a 1 to this bit forces the controller to host mode irrespective of utmiotg_iddig input pin.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Normal Mode</li> <li>■ 1'b1: Force Host Mode</li> </ul> <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 µs is sufficient. This bit is valid only when OTG_MODE = 0, 1 or 2. In all other cases, this bit reads 0.</p>
9	<b>HNP-Capable (HNPCap)</b>	<p>The application uses this bit to control the DWC_otg controller's HNP capabilities.</p> <ul style="list-style-type: none"> <li>■ 1'b0: HNP capability is not enabled.</li> <li>■ 1'b1: HNP capability is enabled.</li> </ul> <p>This bit is writable only if an HNP mode was specified for Mode of Operation in coreConsultant (parameter OTG_MODE). Otherwise, reads return 0.</p>
8	<b>SRP-Capable (SRPCap)</b>	<p>The application uses this bit to control the DWC_otg controller SRP capabilities. If the controller operates as a non-SRP-capable B-device, it cannot request the connected A-device (host) to activate VBUS and start a session.</p> <ul style="list-style-type: none"> <li>■ 1'b0: SRP capability is not enabled.</li> <li>■ 1'b1: SRP capability is enabled.</li> </ul> <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>
6	<b>USB 2.0 High-Speed PHY or USB 1.1 Full-Speed Serial Transceiver Select (PHYSel)</b>	<p>The application uses this bit to select either a high-speed UTMI+ or ULPI PHY, or a full-speed transceiver.</p> <ul style="list-style-type: none"> <li>■ 1'b0: USB 2.0 high-speed UTMI+ or ULPI PHY</li> <li>■ 1'b1: USB 1.1 full-speed serial transceiver</li> </ul> <p>If a USB 1.1 Full-Speed Serial Transceiver interface was not selected in coreConsultant (parameter OTG_FSPHY_INTERFACE = 0), this bit is always 0, with Read Only access.</p> <p>If a high-speed PHY interface was not selected in coreConsultant (parameter OTG_HSPHY_INTERFACE = 0), this bit is always 1, with Read 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>

Bit Number	Register/Bit Name	Description
4	<b>ULPI or UTMI+ Select (ULPI_UTMI_Sel)</b>	<p>The application uses this bit to select either a UTMI+ interface or ULPI Interface.</p> <ul style="list-style-type: none"> <li>■ 1'b0: UTMI+ Interface</li> <li>■ 1'b1: ULPI Interface</li> </ul> <p>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	<b>PHY Interface (PHYIf)</b>	<p>The application uses this bit to configure the controller 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.</p> <ul style="list-style-type: none"> <li>■ 1'b0: 8 bits</li> <li>■ 1'b1: 16 bits</li> </ul> <p>This bit is writable only if UTMI+ and ULPI were selected in coreConsultant configuration (parameter = 2). Otherwise, this bit returns the value for the power-on interface selected during configuration.</p>
	<b>Host Configuration Register (HCFG)</b>	<p>This register configures the controller after power-on. Do not make changes to this register after initializing the host.</p>
23	<b>Enable Scatter/gather DMA in Host mode (DescDMA)</b>	<p>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.</p> <p>NOTE: This bit must be modified only once after a reset. The following combinations are available for programming:</p> <ul style="list-style-type: none"> <li>■ GAHBCFG.DMAEn=0, HCFG.DescDMA=0 =&gt; Slave mode</li> <li>■ GAHBCFG.DMAEn=0, HCFG.DescDMA=1 =&gt; Invalid</li> <li>■ GAHBCFG.DMAEn=1, HCFG.DescDMA=0 =&gt; Buffered DMA mode</li> <li>■ GAHBCFG.DMAEn=1, HCFG.DescDMA=1 =&gt; Scatter/Gather DMA mode</li> </ul> <p>In non Scatter/Gather DMA mode, this bit is reserved.</p>
2	<b>FS- and LS-Only Support (FSLSSupp)</b>	<p>The application uses this bit to control the controller's enumeration speed. Using this bit, the application can make the controller enumerate as a FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.</p> <ul style="list-style-type: none"> <li>■ 1'b0: HS/FS/LS, based on the maximum speed supported by the connected device</li> <li>■ 1'b1: FS/LS-only, even if the connected device can support HS</li> </ul>
	<b>Host Channel-n Characteristics Register (HCCHARn)</b>	

Bit Number	Register/Bit Name	Description
19:18	<b>Endpoint Type (EPType)</b>	Indicates the transfer type selected. <ul style="list-style-type: none"> <li>■ 2'b00: Control</li> <li>■ 2'b01: Isochronous</li> <li>■ 2'b10: Bulk</li> <li>■ 2'b11: Interrupt</li> </ul>
15	<b>Endpoint Direction (EPDir)</b>	Indicates whether the transaction is IN or OUT. <ul style="list-style-type: none"> <li>■ 1'b0: OUT</li> <li>■ 1'b1: IN</li> </ul>
14:11	<b>Endpoint Number (EPNum)</b>	Indicates the endpoint number on the device serving as the data source or sink.
10:0	<b>Maximum Packet Size (MPS)</b>	Indicates the maximum packet size of the associated endpoint.
	<b>Host Channel-n Transfer Size Register (HCTSIZn)</b>	
30:29	<b>PID (Pid)</b>	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. <ul style="list-style-type: none"> <li>■ 2'b00: DATA0</li> <li>■ 2'b01: DATA2</li> <li>■ 2'b10: DATA1</li> <li>■ 2'b11: MDATA (non-control)</li> </ul>
	<b>Device Configuration Register (DCFG)</b>	This register configures the controller in Device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

Bit Number	Register/Bit Name	Description
25:24	<b>Periodic Scheduling Interval (PerSchIntvl)</b>	<p>PerSchIntvl must be programmed only for Scatter/Gather DMA mode.</p> <p><b>Description:</b> 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.</p> <ul style="list-style-type: none"> <li>■ When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data.</li> <li>■ When no periodic endpoints are active, then the internal DMA engine services non-periodic endpoints, ignoring this field.</li> <li>■ After the specified time within a (micro)frame, the DMA switches to fetching for non-periodic endpoints.</li> <li>■ 2'b00: 25% of (micro)frame.</li> <li>■ 2'b01: 50% of (micro)frame.</li> <li>■ 2'b10: 75% of (micro)frame.</li> <li>■ 2'b11: Reserved.</li> </ul>
23	<b>Enable Scatter/Gather DMA in Device mode (DescDMA)</b>	<p>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.</p> <p><b>NOTE:</b> This bit must be modified only once after a reset.</p> <p>The following combinations are available for programming:</p> <ul style="list-style-type: none"> <li>■ GAHBCFG.DMAEn=0,DCFG.DescDMA=0 =&gt; Slave mode</li> <li>■ GAHBCFG.DMAEn=0,DCFG.DescDMA=1 =&gt; Invalid</li> <li>■ GAHBCFG.DMAEn=1,DCFG.DescDMA=0 =&gt; Buffered DMA mode</li> <li>■ GAHBCFG.DMAEn=1,DCFG.DescDMA=1 =&gt; Scatter/Gather DMA mode</li> </ul>
10:4	<b>Device Address (DevAddr)</b>	The application must program this field after every SetAddress control command.
1:0	<b>Device Speed (DevSpd)</b>	<p>Indicates the speed at which the application requires the controller 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 controller is connected. See “Device Initialization” in the Programming Guide for details.</p> <ul style="list-style-type: none"> <li>■ 2'b00: High speed (USB 2.0 PHY clock is 30 MHz or 60 MHz)</li> <li>■ 2'b01: Full speed (USB 2.0 PHY clock is 30 MHz or 60 MHz)</li> <li>■ 2'b10: Reserved</li> <li>■ 2'b11: Full speed (USB 1.1 transceiver clock is 48 MHz)</li> </ul>
	<b>Device Endpoint-n Control Register (DIEPCTLn/DOEPCTLn)</b>	The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Bit Number	Register/Bit Name	Description
29	<b>Set DATA1 PID (SetD1PID)</b>	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.
28	<b>Set DATA0 PID (SetD0PID)</b>	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.
19:18	<b>Endpoint Type (EPType)</b>	Applies to IN and OUT endpoints. This is the transfer type supported by this logical endpoint. <ul style="list-style-type: none"> <li>■ 2'b00: Control</li> <li>■ 2'b01: Isochronous</li> <li>■ 2'b10: Bulk</li> <li>■ 2'b11: Interrupt</li> </ul>
10:00	<b>Maximum Packet Size (MPS)</b>	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.

## 6.4 Additional Details on GLPMCFG.HIRD\_Thres Register Field

[Table 6-2](#) shows host-mode resume signaling time based on HIRD\_Thres register field value.

**Table 6-2 HIRD\_Thres Register Field and Host-Mode Resume Signaling Time**

Thres[3:0]	Host Mode Resume Signaling Time (μs)	
	EnBESL = 1'b0	EnBESL = 1'b1
4'b0000	60	75
4'b0001	135	100
4'b0010	210	150
4'b0011	285	250
4'b0100	360	350
4'b0101	435	450
4'b0110	510	950
4'b0111	585	Invalid
4'b1000	660	Invalid
4'b1001	735	Invalid
4'b1010	810	Invalid
4'b1011	885	Invalid
4'b1100	960	Invalid
4'b1101	Invalid	Invalid
4'b1110	Invalid	Invalid
4'b1111	Invalid	Invalid

[Table 6-3](#) shows the difference in behavior between the UTMI and ULPI interface in different modes of operation:

**Table 6-3 Difference in UTMI and ULPI Behavior in Different Modes of Operation**

Bit 7	Bit 6	sleep_n	I1_suspend_n	suspend_n	Mode of Operation
0	1	1	1	1	Normal Operation
0	0	1	1	0	L2 Suspend
1	0	1	0	1	L1 Deep Sleep
1	1	0	1	1	L1 Shallow Sleep



# Register Descriptions

---

This chapter details all possible registers in the IP. They are arranged hierarchically into maps and blocks (banks). Your actual configuration might not contain all of these registers.

**Attention: For configurable IP titles, do not use this document to determine the exact attributes of your register map. It is for reference purposes only.**

When you configure the controller in coreConsultant, you must access the register attributes for your actual configuration at workspace/report/ComponentRegisters.html or workspace/report/ComponentRegisters.xml after you have completed the report creation activity. That report comes from the exact same source as this chapter but removes all the registers that are not in your actual configuration. This does not apply to non-configurable IP titles. In addition, all parameter expressions are evaluated to actual values. Therefore, the Offset and Memory Access values might change depending on your actual configuration.

Some expressions might refer to TCL functions or procedures (sometimes identified as <functionof>) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the controller in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

## Exists Expressions

These expressions indicate the combination of configuration parameters required for a register, field, or block to exist in the memory map. The expression is only valid in the local context and does not indicate the conditions for existence of the parent. For example, the expression for a bit field in a register assumes that the register exists and does not include the conditions for existence of the register.

## Offset

The term *Offset* is synonymous with *Address*.

## Memory Access Attributes

The Memory Access attribute is defined as <ReadBehavior>/<WriteBehavior> which are defined in the following table.

**Table 7-1 Possible Read and Write Behaviors**

<b>Read (or Write) Behavior</b>	<b>Description</b>
RC	A read clears this register field.
RS	A read sets this register field.
RM	A read modifies the contents of this register field.
Wo	You can only write once to this register field.
W1C	A write of 1 clears this register field.
W1S	A write of 1 sets this register field.
W1T	A write of 1 toggles this register field.
W0C	A write of 0 clears this register field.
W0S	A write of 0 sets this register field.
W0T	A write of 0 toggles this register field.
WC	Any write clears this register field.
WS	Any write sets this register field.
WM	Any write toggles this register field.
no Read Behavior attribute	You cannot read this register. It is Write-Only.
no Write Behavior attribute	You cannot write to this register. It is Read-Only.

**Table 7-2 Memory Access Examples**

<b>Memory Access</b>	<b>Description</b>
R	Read-only register field.
W	Write-only register field.
R/W	Read/write register field.
R/W1C	You can read this register field. Writing 1 clears it.
RC/W1C	Reading this register field clears it. Writing 1 clears it.
R/Wo	You can read this register field. You can only write to it once.

### Special Optional Attributes

Some register fields might use the following optional attributes.

**Table 7-3 Optional Attributes**

<b>Attribute</b>	<b>Description</b>
Volatile	As defined by the IP-XACT specification. If true, indicates in the case of a write followed by read, or in the case of two consecutive reads, there is no guarantee as to what is returned by the read on the second transaction or that this return value is consistent with the write or read of the first transaction. The element implies there is some additional mechanism by which this field can acquire new values other than by reads/writes/resets and other access methods known to IP-XACT. For example, when the core updates the register field contents.
Testable	As defined by the IP-XACT specification. Possible values are unconstrained, untestable, readOnly, writeAsRead, restore. Untestable means that this field is untestable by a simple automated register test. For example, the read-write access of the register is controlled by a pin or another register. readOnly means that you should not write to this register; only read from it. This might apply for a register that modifies the contents of another register.
Reset Mask	As defined by the IP-XACT specification. Indicates that this register field has an unknown reset value. For example, the reset value is set by another register or an input pin; or the register is implemented using RAM.
* Varies	Indicates that the memory access (or reset) attribute (read, write behavior) is not fixed. For example, the read-write access of the register is controlled by a pin or another register. Or when the access depends on some configuration parameter; in this case the post-configuration report in coreConsultant gives the actual access value.

**Note:**

For some register fields, the value of reset is shown as "*\* Varies based on Configuration*". For these register fields, the value after reset depends on the value of the configuration parameters that you chose during configuration. For the reset value specific to your configuration, see coreConsultant ComponentRegisters report.

**Component Banks/Blocks**

The following table shows the address blocks for each memory map. Follow the link for an address block to see a table of its registers.

**Table 7-4 Address Banks/Blocks for Memory Map: DWC\_otg\_map**

Address Block	Description
DWC_otg_intreg on <a href="#">page 302</a>	Internal register map <b>Exists:</b> Always
DWC_otg_DFIFO_0	Data FIFO Access Register Map 0 <b>Exists:</b> Always
DWC_otg_DFIFO_1	Data FIFO Access Register Map 1 <b>Exists:</b> Always
DWC_otg_DFIFO_2	Data FIFO Access Register Map 2 <b>Exists:</b> Always
DWC_otg_DFIFO_3	Data FIFO Access Register Map 3 <b>Exists:</b> Always
DWC_otg_DFIFO_4	Data FIFO Access Register Map 4 <b>Exists:</b> Always
DWC_otg_DFIFO_5	Data FIFO Access Register Map 5 <b>Exists:</b> Always
DWC_otg_DFIFO_6	Data FIFO Access Register Map 6 <b>Exists:</b> Always
DWC_otg_DFIFO_7	Data FIFO Access Register Map 7 <b>Exists:</b> Always
DWC_otg_DFIFO_8	Data FIFO Access Register Map 8 <b>Exists:</b> Always
DWC_otg_DFIFO_9	Data FIFO Access Register Map 9 <b>Exists:</b> Always
DWC_otg_DFIFO_10	Data FIFO Access Register Map 10 <b>Exists:</b> Always
DWC_otg_DFIFO_11	Data FIFO Access Register Map 11 <b>Exists:</b> Always
DWC_otg_DFIFO_12	Data FIFO Access Register Map 12 <b>Exists:</b> Always
DWC_otg_DFIFO_13	Data FIFO Access Register Map 13 <b>Exists:</b> Always
DWC_otg_DFIFO_14	Data FIFO Access Register Map 14 <b>Exists:</b> Always

**Table 7-4 Address Banks/Blocks for Memory Map: DWC\_otg\_map (Continued)**

Address Block	Description
DWC_otg_DFIFO_15	Data FIFO Access Register Map 15 <b>Exists:</b> Always
DWC_otg_DFIFO_Direct_access	Data FIFO Direct Access Register Map <b>Exists:</b> Always

## 7.1 DWC\_otg\_map/DWC\_otg\_intreg Registers

Internal register map. Follow the link for the register to see a detailed description of the register.

**Table 7-5 Registers for Address Block: DWC\_otg\_map/DWC\_otg\_intreg**

Register	Offset	Description
GOTGCTL on <a href="#">page 307</a>	0x0	Control and Status Register
GOTGINT on <a href="#">page 319</a>	0x4	Interrupt Register
GAHBCFG on <a href="#">page 323</a>	0x8	AHB Configuration Register
GUSBCFG on <a href="#">page 332</a>	0xc	USB Configuration Register
GRSTCTL on <a href="#">page 346</a>	0x10	Reset Register
GINTSTS on <a href="#">page 355</a>	0x14	Interrupt Register
GINTMSK on <a href="#">page 371</a>	0x18	Interrupt Mask Register
GRXSTSR on <a href="#">page 379</a>	0x1c	Receive Status Debug Read Register
GRXSTSP on <a href="#">page 384</a>	0x20	Receive Status Read/Pop Register
GRXFSSIZ on <a href="#">page 388</a>	0x24	Receive FIFO Size Register
GNPTXFSIZ on <a href="#">page 389</a>	0x28	Non-periodic Transmit FIFO Size Register
GNPTXSTS on <a href="#">page 392</a>	0x2c	Non-periodic Transmit FIFO/Queue Status Register
GI2CCTL on <a href="#">page 395</a>	0x30	I2C Access Register
GPVNDCTL on <a href="#">page 399</a>	0x34	PHY Vendor Control Register
GGPIO on <a href="#">page 402</a>	0x38	General Purpose Input/Output Register
GUID on <a href="#">page 403</a>	0x3c	User ID Register
GSNPSID on <a href="#">page 404</a>	0x40	Synopsys ID Register
GHWCFG1 on <a href="#">page 405</a>	0x44	User Hardware Configuration 1 Register
GHWCFG2 on <a href="#">page 406</a>	0x48	User Hardware Configuration 2 Register
GHWCFG3 on <a href="#">page 414</a>	0x4c	User Hardware Configuration 3 Register
GHWCFG4 on <a href="#">page 420</a>	0x50	User Hardware Configuration 4 Register
GLPMCFG on <a href="#">page 429</a>	0x54	LPM Config Register
GPWRDN on <a href="#">page 445</a>	0x58	Global Power Down Register
GDFIFO CFG on <a href="#">page 456</a>	0x5c	Global DFIFO Configuration Register
GADPCTL on <a href="#">page 457</a>	0x60	ADP Timer, Control and Status Register

**Table 7-5 Registers for Address Block: DWC\_otg\_map/DWC\_otg\_intreg (Continued)**

<b>Register</b>	<b>Offset</b>	<b>Description</b>
GREFCLK on <a href="#">page 463</a>	0x64	ref_clk Control Register
GINTMSK2 on <a href="#">page 465</a>	0x68	Interrupt Mask Register 2
GINTSTS2 on <a href="#">page 466</a>	0x6c	Interrupt Register 2
HPTXFSIZ on <a href="#">page 467</a>	0x100	Host Periodic Transmit FIFO Size Register
DPTXFSIZi (for i = 1; i <= OTG_NUM_PERIO_EPS-1) on <a href="#">page 469</a>	0x104 + (i-1)*004	Device Periodic Transmit FIFO-i Size Register
DIEPTXF <sub>i</sub> (for i = 1; i <= OTG_NUM_IN_EPS-1) on <a href="#">page 471</a>	0x104 + (i-1)*004	Device IN Endpoint Transmit FIFO Size Register
HCFG on <a href="#">page 474</a>	0x400	Host Configuration Register
HFIR on <a href="#">page 481</a>	0x404	Host Frame Interval Register
HFNUM on <a href="#">page 483</a>	0x408	Host Frame Number/Frame Time Remaining Register
HPTXSTS on <a href="#">page 485</a>	0x410	Host Periodic Transmit FIFO/Queue Status Register
HAINT on <a href="#">page 488</a>	0x414	Host All Channels Interrupt Register
HAINTMSK on <a href="#">page 489</a>	0x418	Host All Channels Interrupt Mask Register
HFLBAddr on <a href="#">page 490</a>	0x41c	Host Frame List Base Address Register
HPRT on <a href="#">page 491</a>	0x440	Host Port Control and Status Register
HCCHAR <sub>i</sub> (for i = 0; i <= OTG_NUM_HOST_CHAN-1) on <a href="#">page 500</a>	0x500 + i*20	Host Channel Characteristics Register
HCSPLT <sub>i</sub> (for i = 0; i <= OTG_NUM_HOST_CHAN-1) on <a href="#">page 506</a>	0x504 + i*20	Host Channel Split Control Register
HCINT <sub>i</sub> (for i = 0; i <= OTG_NUM_HOST_CHAN-1) on <a href="#">page 509</a>	0x508 + i*20	Host Channel Interrupt Register
HCINTMSK <sub>i</sub> (for i = 0; i <= OTG_NUM_HOST_CHAN-1) on <a href="#">page 515</a>	0x50C + i*20	Host Channel Interrupt Mask Register
HCTSIZ <sub>i</sub> (for i = 0; i <= OTG_NUM_HOST_CHAN-1) on <a href="#">page 519</a>	0x510 + i*20	Host Channel Transfer Size Register

**Table 7-5 Registers for Address Block: DWC\_otg\_map/DWC\_otg\_intreg (Continued)**

<b>Register</b>	<b>Offset</b>	<b>Description</b>
HCDMAi (for i = 0; i <= OTG_NUM_HOST_CHAN-1) on <a href="#">page 523</a>	0x514 + i*20	Host Channel DMA Address Register
HCDMABi (for i = 0; i <= OTG_NUM_HOST_CHAN-1) on <a href="#">page 526</a>	0x51C + i*20	Host Channel DMA Buffer Address Register
DCFG on <a href="#">page 527</a>	0x800	Device Configuration Register
DCTL on <a href="#">page 534</a>	0x804	Device Control Register
DSTS on <a href="#">page 544</a>	0x808	Device Status Register
DIEPMSK on <a href="#">page 547</a>	0x810	Device IN Endpoint Common Interrupt Mask Register
DOEPMSK on <a href="#">page 550</a>	0x814	Device OUT Endpoint Common Interrupt Mask Register
DAINT on <a href="#">page 554</a>	0x818	Device All Endpoints Interrupt Register
DAINTMSK on <a href="#">page 560</a>	0x81c	Device All Endpoints Interrupt Mask Register
DTKNQR1 on <a href="#">page 566</a>	0x820	Device IN Token Sequence Learning Queue Read Register 1
DTKNQR2 on <a href="#">page 568</a>	0x824	Device IN Token Sequence Learning Queue Read Register 2
DVBUSDIS on <a href="#">page 569</a>	0x828	Device VBUS Discharge Time Register
DVBUSPULSE on <a href="#">page 570</a>	0x82c	Device VBUS Pulsing Time Register
DTHRCTL on <a href="#">page 571</a>	0x830	Device Threshold Control Register
DTKNQR3 on <a href="#">page 575</a>	0x830	Device IN Token Sequence Learning Queue Read Register 3
DIEPEMPMSK on <a href="#">page 576</a>	0x834	Device IN Endpoint FIFO Empty Interrupt Mask Register
DTKNQR4 on <a href="#">page 578</a>	0x834	Device IN Token Sequence Learning Queue Read Register 4
DEACHINT on <a href="#">page 579</a>	0x838	Device Each Endpoints Interrupt Register
DEACHINTMSK on <a href="#">page 582</a>	0x83c	Device Each Endpoints Interrupt Mask Register
DIEPEACHMSK0 on <a href="#">page 588</a>	0x840	Device Each IN Endpoint 0 Interrupt Register
DIEPEACHMSKi (for i = 1; i <= OTG_NUM_EPS) on <a href="#">page 591</a>	0x840 + i*004	Device Each IN Endpoint Interrupt Register
DOEPEACHMSK0 on <a href="#">page 594</a>	0x880	Device Each OUT Endpoint 0 Interrupt Register
DOEPEACHMSKi (for i = 1; i <= OTG_NUM_EPS) on <a href="#">page 598</a>	0x880 + i*004	Device Each OUT Endpoint Interrupt Register

**Table 7-5 Registers for Address Block: DWC\_otg\_map/DWC\_otg\_intreg (Continued)**

<b>Register</b>	<b>Offset</b>	<b>Description</b>
DIEPCTL0 on <a href="#">page 602</a>	0x900	Device Control IN Endpoint 0 Control Register
DIEPINT0 on <a href="#">page 608</a>	0x908	Device IN Endpoint 0 Interrupt Register
DIEPTSIZ0 on <a href="#">page 614</a>	0x910	Device IN Endpoint 0 Transfer Size Register
DIEPDMA0 on <a href="#">page 616</a>	0x914	Device IN Endpoint 0 DMA Address Register
DTXFSTS0 on <a href="#">page 617</a>	0x918	Device IN Endpoint Transmit FIFO Status Register 0
DIEPDMA0 on <a href="#">page 619</a>	0x91c	Device IN Endpoint 0 Buffer Address Register
DIEPCTL $i$ (for $i = 1; i <= \text{OTG\_NUM\_EPS}$ ) on <a href="#">page 620</a>	0x900 + $i*20$	Device Control IN Endpoint Control Register
DIEPINT $i$ (for $i = 1; i <= \text{OTG\_NUM\_EPS}$ ) on <a href="#">page 630</a>	0x908 + $i*20$	Device IN Endpoint Interrupt Register
DIEPTSIZ $i$ (for $i = 1; i <= \text{OTG\_NUM\_EPS}$ ) on <a href="#">page 636</a>	0x910 + $i*20$	Device IN Endpoint Transfer Size Register
DIEPDMA $i$ (for $i = 1; i <= \text{OTG\_NUM\_EPS}$ ) on <a href="#">page 638</a>	0x914 + $i*20$	Device IN Endpoint DMA Address Register
DTXFSTS $i$ (for $i = 1; i <= \text{OTG\_NUM\_EPS}$ ) on <a href="#">page 639</a>	0x918 + $i*20$	Device IN Endpoint Transmit FIFO Status Register
DIEPDMA $i$ (for $i = 1; i <= \text{OTG\_NUM\_EPS}$ ) on <a href="#">page 641</a>	0x91C + $i*20$	Device IN Endpoint Buffer Address Register
DOEPCTL0 on <a href="#">page 642</a>	0xb00	Device Control OUT Endpoint 0 Control Register
DOEPINT0 on <a href="#">page 647</a>	0xb08	Device OUT Endpoint 0 Interrupt Register
DOEPTSIZ0 on <a href="#">page 653</a>	0xb10	Device OUT Endpoint 0 Transfer Size Register
DOEPDMA0 on <a href="#">page 655</a>	0xb14	Device OUT Endpoint 0 DMA Address Register
DOEPDMAB0 on <a href="#">page 656</a>	0xb1c	Device OUT Endpoint 0 DMA Buffer Address Register
DOEPCTL $i$ (for $i = 1; i <= \text{OTG\_NUM\_EPS}$ ) on <a href="#">page 657</a>	0xB00 + $i*20$	Device Control OUT Endpoint Control Register

**Table 7-5 Registers for Address Block: DWC\_otg\_map/DWC\_otg\_intreg (Continued)**

<b>Register</b>	<b>Offset</b>	<b>Description</b>
DOEPINT $i$ (for $i = 1; i \leq OTG\_NUM\_EPS$ ) on <a href="#">page 666</a>	0xB08 + $i*20$	Device OUT Endpoint Interrupt Register
DOEPTSIZ $i$ (for $i = 1; i \leq OTG\_NUM\_EPS$ ) on <a href="#">page 672</a>	0xB10 + $i*20$	Device OUT Endpoint Transfer Size Register
DOEPDMA $i$ (for $i = 1; i \leq OTG\_NUM\_EPS$ ) on <a href="#">page 674</a>	0xB14 + $i*20$	Device OUT Endpoint DMA Address Register
DOEPDMAB $i$ (for $i = 1; i \leq OTG\_NUM\_EPS$ ) on <a href="#">page 675</a>	0xB1C + $i*20$	Device OUT Endpoint Buffer Address Register
PCGCCTL on <a href="#">page 676</a>	0xe00	Power and Clock Gating Control Register
PCGCCTL1 on <a href="#">page 684</a>	0xe04	Power and Clock Gating Control Register1

### 7.1.1 GOTGCTL

- **Name:** Control and Status Register
  - **Description:** The OTG Control and Status register controls the behavior and reflects the status of the OTG function of the controller.
  - **Size:** 32 bits
  - **Offset:** 0x0
  - **Exists:** Always

Testmode_corr_eUSB2	31
Reserved_30_28	30:28
ChirpEn	27
MultValldBC	26:22
CurlMod	21
OTGVer	20
BSesVid	19
ASesVid	18
Dbnctime	17
ComIDSts	16
DbncestBypass	15
Rsvd	14:13
EHEn	12
DevHNPEn	11
HstSetHNPEn	10
HNPRreq	9
HstNegScs	8
BvalidOvVal	7
BvalidOvEn	6
AvalidOvVal	5
AvalidOvEn	4
VbvalidOvVal	3
VbvalidOvEn	2
SesReq	1
SesReqScs	0

**Table 7-6 Fields for Register: GOTGCTL**

Bits	Name	Memory Access	Description
31	Testmode_corr_eUSB2	R/W	<p>UTMI IF correction for eUSB2 PHY during Test mode</p> <p>This bit is used to modify the behavior of UTMI 8-bit interface signals during test J and test K sequences when eUSB2 PHY is used.</p> <p>When this bit is set to 1'b1, the controller asserts utmi_txvalid and utmi_opmode in the same cycle during test J or test K sequence execution.</p> <p><b>Note:</b> This bit is applicable only if eUSB2 PHY is used with 8-bit UTMI interface.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (eUSB2_corr_disable): The controller asserts utmi_txvalid one cycle later than utmi_opmode.</li> <li>■ 0x1 (eUSB2_corr_enable): The controller asserts utmi_txvalid and utmi_opmode in the same cycle.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_HSPHY_DWIDTH==0</p>

**Table 7-6 Fields for Register: GOTGCTL (Continued)**

Bits	Name	Memory Access	Description
30:28	Reserved_30_28	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
27	ChirpEn	R/W	<p>Mode: Device Only</p> <p>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. Do not set this bit when core is operating in HSIC mode because HSIC always operates at High Speed and High speed chirp is not used</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (CHIRP_DISABLE): The controller does not assert chirp_on before sending an actual Chirp "K" signal on USB.</li> <li>■ 0x1 (CHIRP_ENABLE): The controller asserts chirp_on before sending an actual Chirp "K" signal on USB.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6) &amp;&amp; OTG_BC_SUPPORT==1</p>
26:22	MultValldBC	R	<p>Mode: Host and Device</p> <p>Multi Valued ID pin (MultValldBC)</p> <p>Battery Charger ACA inputs in the following order:</p> <ul style="list-style-type: none"> <li>■ Bit 26: rid_float.</li> <li>■ Bit 25: rid_gnd</li> <li>■ Bit 24: rid_a</li> <li>■ Bit 23: rid_b</li> <li>■ Bit 22: rid_c</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (RID_C): B-Device connected to ACA. VBUS is on.</li> <li>■ 0x2 (RID_B): B-Device connected to ACA. VBUS is off.</li> <li>■ 0x4 (RID_A): A-Device connected to ACA</li> <li>■ 0x8 (RID_GND): A-Device not connected to ACA</li> <li>■ 0x10 (RID_FLOAT): B-Device not connected to ACA</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_BC_SUPPORT==1</p>

**Table 7-6 Fields for Register: GOTGCTL (Continued)**

Bits	Name	Memory Access	Description
21	CurMod	R	<p>Current Mode of Operation (CurMod) Mode: Host and Device Indicates the current mode.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Device mode</li> <li>■ 1'b1: Host mode</li> </ul> <p><b>Reset:</b></p> <ul style="list-style-type: none"> <li>■ 1'b1 in Host-only mode (OTG_MODE=5 or 6)</li> <li>■ 1'b0 in all other configurations</li> </ul> <p><b>Note:</b> The reset value of this register field can be read only after the PHY clock is stable, or if IDDIG_FILTER is enabled, wait for the filter timer to expire to read the correct reset value which ever event is later.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DEVICEMODE): Current mode is device mode.</li> <li>■ 0x1 (HOSTMODE): Current mode is host mode.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
20	OTGVer	R/W	<p>OTG Version (OTGVer) Indicates the OTG revision.</p> <ul style="list-style-type: none"> <li>■ 1'b0: OTG Version 1.3. In this version the core supports Data line pulsing and VBus pulsing for SRP.</li> <li>■ 1'b1: OTG Version 2.0. In this version the core supports only Data line pulsing for SRP.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (VER13): Supports OTG Version 1.3</li> <li>■ 0x1 (VER20): Supports OTG Version 2.0</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-6 Fields for Register: GOTGCTL (Continued)**

Bits	Name	Memory Access	Description
19	BSesVld	R	<p>Mode: Device only B-Session Valid (BSesVld) Indicates the Device mode transceiver status.</p> <ul style="list-style-type: none"> <li>■ 1'b0: B-session is not valid.</li> <li>■ 1'b1: B-session is valid.</li> </ul> <p>In OTG mode, you can use this bit to determine if the device is connected or disconnected.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>■ 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.</li> <li>■ In case of OTG_MODE=0, the reset value of this bit is 1'b0.</li> <li>■ The reset value of this register field can be read only after the PHY clock is stable, or if IDDIG_FILTER is enabled, wait for the filter timer to expire to read the correct reset value which ever event is later.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTVALID): B-session is not valid.</li> <li>■ 0x1 (VALID): B-session is valid.</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6</p>
18	ASesVld	R	<p>Mode: Host only A-Session Valid (ASesVld) Indicates the Host mode transceiver status.</p> <ul style="list-style-type: none"> <li>■ 1'b0: A-session is not valid</li> <li>■ 1'b1: A-session is valid</li> </ul> <p><b>Note:</b> If you do not enabled 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> <p>In case of OTG_MODE=0, the reset value of this bit is 1'b0.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTVALID): A-session is not valid.</li> <li>■ 0x1 (VALID): A-session is valid.</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> OTG_MODE == 0    OTG_MODE == 1    OTG_MODE == 2    OTG_MODE == 5    OTG_MODE == 6</p> <p><b>Volatile:</b> true</p>

**Table 7-6 Fields for Register: GOTGCTL (Continued)**

Bits	Name	Memory Access	Description
17	DbncTime	R	<p>Mode: Host only          Long/Short Debounce Time (DbncTime)          Indicates the debounce time of a detected connection.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Long debounce time, used for physical connections (100 ms + 2.5 micro-sec)</li> <li>■ 1'b1: Short debounce time, used for soft connections (2.5 micro-sec)</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (LONG): Long debounce time, used for physical connections (100 ms + 2.5 micro-sec)</li> <li>■ 0x1 (SHORT): Short debounce time, used for soft connections (2.5 micro-sec)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE == 0    OTG_MODE == 1    OTG_MODE == 2    OTG_MODE == 5    OTG_MODE == 6</p>
16	ConIDsts	R	<p>Mode: Host and Device          Connector ID Status (ConIDsts)          Indicates the connector ID status on a connect event.</p> <ul style="list-style-type: none"> <li>■ 1'b0: The core is in A-Device mode.</li> <li>■ 1'b1: The core is in B-Device mode.</li> </ul> <p><b>Note:</b> The reset value of this register field can be read only after the PHY clock is stable, or if IDDIG_FILTER is enabled, wait for the filter timer to expire to read the correct reset value which ever event is later. <b>Reset:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: in host only mode (OTG_MODE = 5 or 6)</li> <li>■ 1'b1: in all other configurations</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MODEA): The core is in A-Device mode.</li> <li>■ 0x1 (MODEB): The core is in B-Device mode.</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>

**Table 7-6 Fields for Register: GOTGCTL (Continued)**

Bits	Name	Memory Access	Description
15	DbnceFltrBypass	R/W	<p>Mode: Host and Device Debounce Filter Bypass Bypass Debounce filters for avalid, bvalid, vbusvalid, sessend, iddig signals when enabled.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Disabled</li> <li>■ 1'b1: Enabled</li> </ul> <p><b>Note:</b> This register bit is valid only when debounce filters are present in core.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Debounce Filter Bypass is disabled.</li> <li>■ 0x1 (ENABLED): Debounce Filter Bypass is enabled.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_IDDIG_FILTER == 1    OTG_EN_VBUSVALID_FILTER == 1    OTG_EN_A_VALID_FILTER == 1    OTG_EN_B_VALID_FILTER == 1    OTG_EN_SESSIONEND_FILTER == 1</p>
14:13			<b>Reserved Field:</b> Yes
12	EHEn	R/W	<p>Mode: SRP Capable Host Embedded Host Enable (EHEn) It is used to select between OTG A Device state Machine and Embedded Host state machine.</p> <ul style="list-style-type: none"> <li>■ 1'b0: OTG A Device state machine is selected</li> <li>■ 1'b1: Embedded Host State Machine is selected</li> </ul> <p><b>Note:</b> This field is valid only in SRP-Capable OTG Mode (OTG_MODE=0,1).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (EH_DISABLE): OTG A Device state machine is selected.</li> <li>■ 0x1 (EH_ENABLE): Embedded Host state machine is selected.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE == 0    OTG_MODE == 1</p>

**Table 7-6 Fields for Register: GOTGCTL (Continued)**

Bits	Name	Memory Access	Description
11	DevHNPEn	R/W	<p>Mode: HNP Capable OTG Device Device HNP Enabled (DevHNPEn) The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host.</p> <ul style="list-style-type: none"> <li>■ 1'b0: HNP is not enabled in the application</li> <li>■ 1'b1: HNP is enabled in the application</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): HNP is disabled</li> <li>■ 0x1 (ENABLED): HNP is enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE == 0</p>
10	HstSetHNPEn	R/W	<p>Mode: HNP Capable OTG Host Host Set HNP Enable (HstSetHNPEn) The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Host Set HNP is not enabled</li> <li>■ 1'b1: Host Set HNP is enabled</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Host Set HNP is not enabled.</li> <li>■ 0x1 (ENABLED): Host Set HNP is enabled.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE == 0</p>

**Table 7-6 Fields for Register: GOTGCTL (Continued)**

Bits	Name	Memory Access	Description
9	HNPReq	R/W	<p>Mode: HNP Capable OTG Device HNP Request (HNPReq)</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 controller clears this bit when the HstNegSucStsChng bit is cleared.</p> <ul style="list-style-type: none"> <li>■ 1'b0: No HNP request</li> <li>■ 1'b1: HNP request</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No HNP request</li> <li>■ 0x1 (ENABLED): HNP request</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE == 0</p>
8	HstNegScs	R	<p>Mode: HNP-capable Device Host Negotiation Success (HstNegScs)</p> <p>The controller sets this bit when host negotiation is successful. The controller clears this bit when the HNP Request (HNPReq) bit in this register is set.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Host negotiation failure</li> <li>■ 1'b1: Host negotiation success</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (FAIL): Host negotiation failure</li> <li>■ 0x1 (SUCCESS): Host negotiation success</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE == 0</p>

**Table 7-6 Fields for Register: GOTGCTL (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
7	BvalidOvVal	R/W	<p>Mode: Device only  B-Peripheral Session Valid OverrideValue (BvalidOvVal)  This bit is used to set Override value for Bvalid signal when GOTGCTL.BvalidOvEn is set.</p> <ul style="list-style-type: none"> <li>■ 1'b0 : Bvalid value is 1'b0 when GOTGCTL.BvalidOvEn =1</li> <li>■ 1'b1 : Bvalid value is 1'b1 when GOTGCTL.BvalidOvEn =1</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (VALUE0): Bvalid value when GOTGCTL.BvalidOvEn =1</li> <li>■ 0x1 (VALUE1): Bvalid value when GOTGCTL.BvalidOvEn =1</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6</p>
6	BvalidOvEn	R/W	<p>Mode: Device only  B-Peripheral Session Valid Override Value (BvalidOvEn)  This bit is used to enable/disable the software to override the Bvalid signal using the GOTGCTL.BvalidOvVal.</p> <ul style="list-style-type: none"> <li>■ 1'b1 : Internally Bvalid received from the PHY is overridden with GOTGCTL.BvalidOvVal.</li> <li>■ 1'b0 : Override is disabled and bvalid signal from the respective PHY selected is used internally by the force</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Override is disabled and bvalid signal from the respective PHY selected is used internally by the core</li> <li>■ 0x1 (ENABLED): Internally Bvalid received from the PHY is overridden with GOTGCTL.BvalidOvVal</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6</p>

**Table 7-6 Fields for Register: GOTGCTL (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
5	AvalidOvVal	R/W	<p>Mode: Host only  A-Peripheral Session Valid OverrideValue (AvalidOvVal)  This bit is used to set Override value for Avalid signal when GOTGCTL.AvalidOvEn is set.</p> <ul style="list-style-type: none"> <li>■ 1'b0 : Avalid value is 1'b0 when GOTGCTL.AvalidOvEn =1</li> <li>■ 1'b1 : Avalid value is 1'b1 when GOTGCTL.AvalidOvEn =1</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (VALUE0): Avalid value is 1'b0 when GOTGCTL.AvalidOvEn =1</li> <li>■ 0x1 (VALUE1): Avalid value is 1'b1 when GOTGCTL.AvalidOvEn =1</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_MODE != 3 &amp;&amp; OTG_MODE != 4</p>
4	AvalidOvEn	R/W	<p>Mode: Host only  A-Peripheral Session Valid Override Enable (AvalidOvEn)  This bit is used to enable/disable the software to override the Avalid signal using the GOTGCTL.AvalidOvVal.</p> <ul style="list-style-type: none"> <li>■ 1'b1: Internally Avalid received from the PHY is overridden with GOTGCTL.AvalidOvVal.</li> <li>■ 1'b0: Override is disabled and avalid signal from the respective PHY selected is used internally by the core</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Derive AValid from PHY</li> <li>■ 0x1 (ENABLED): Derive Avalid from GOTGCTL.AvalidOvVal</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_MODE != 3 &amp;&amp; OTG_MODE != 4</p>

**Table 7-6 Fields for Register: GOTGCTL (Continued)**

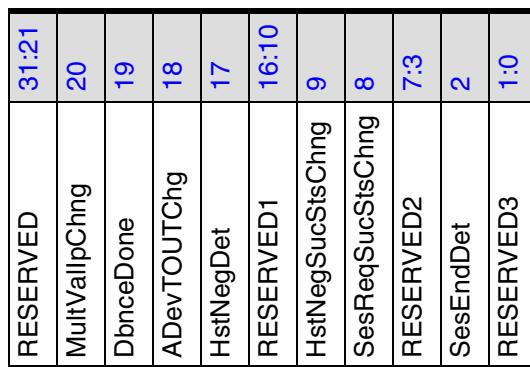
Bits	Name	Memory Access	Description
3	VbvalidOvVal	R/W	<p>Mode: Host only            VBUS Valid OverrideValue (VbvalidOvVal)            This bit is used to set Override value for vbusvalid signal when GOTGCTL.VbvalidOvEn is set.</p> <ul style="list-style-type: none"> <li>■ 1'b0 : vbusvalid value is 1'b0 when GOTGCTL.VbvalidOvEn =1</li> <li>■ 1'b1 : vbusvalid value is 1'b1 when GOTGCTL.VbvalidOvEn =1</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (SET0): vbusvalid value when GOTGCTL.VbvalidOvEn = 1</li> <li>■ 0x1 (SET1): vbusvalid value when GOTGCTL.VbvalidOvEn is 1</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 3 &amp;&amp; OTG_MODE != 4</p>
2	VbvalidOvEn	R/W	<p>Mode: Host only            VBUS Valid Override Enable (VbvalidOvEn)            This bit is used to enable/disable the software to override the Bvalid signal using the GOTGCTL.VbvalidOvVal.</p> <ul style="list-style-type: none"> <li>■ 1'b1 : Internally Bvalid received from the PHY is overridden with GOTGCTL.VbvalidOvVal.</li> <li>■ 1'b0 : Override is disabled and bvalid signal from the respective PHY selected is used internally by the controller.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Override is disabled and bvalid signal from the respective PHY selected is used internally by the controller</li> <li>■ 0x1 (ENABLED): The vbus-valid signal received from the PHY is overridden with GOTGCTL.VbvalidOvVal</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 3 &amp;&amp; OTG_MODE != 4</p>

**Table 7-6 Fields for Register: GOTGCTL (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
1	SesReq	R/W	<p>Mode: Device only Session Request (SesReq) 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.</p> <p>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> <ul style="list-style-type: none"> <li>■ 1'b0: No session request</li> <li>■ 1'b1: Session request</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOREQUEST): No session request</li> <li>■ 0x1 (REQUEST): Session request</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE == 0    OTG_MODE == 1    OTG_MODE == 3</p>
0	SesReqScs	R	<p>Mode: Device only Session Request Success (SesReqScs) The core sets this bit when a session request initiation is successful.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Session request failure</li> <li>■ 1'b1: Session request success</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (FAIL): Session request failure</li> <li>■ 0x1 (SUCCESS): Session request success</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE == 0    OTG_MODE == 1    OTG_MODE == 3</p>

### 7.1.2 GOTGINT

- **Name:** Interrupt Register
- **Description:** The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.
- **Size:** 32 bits
- **Offset:** 0x4
- **Exists:** Always



**Table 7-7 Fields for Register: GOTGINT**

Bits	Name	Memory Access	Description
31:21	RESERVED	R	<b>RESERVED</b> <b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead
20	MultVallpChng	R/W1C	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. <b>Values:</b> <ul style="list-style-type: none"> <li>■ 0x0 (NO_ACA_PIN_CHANGE): Indicates there is no change in ACA pin value</li> <li>■ 0x1 (ACA_PIN_CHANGE): Indicates there is a change in ACA pin value</li> </ul> <b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_BC_SUPPORT==1 <b>Testable:</b> writeAsRead

**Table 7-7 Fields for Register: GOTGINT (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
19	DbnceDone	R/W1C	<p>Mode: Host only Debounce Done (DbnceDone) 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). This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): After Connect waiting for Debounce to complete</li> <li>■ 0x1 (ACTIVE): Debounce completed</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE == 0    OTG_MODE == 1    OTG_MODE == 2    OTG_MODE == 5    OTG_MODE == 6</p>
18	ADevTOUTChg	R/W1C	<p>Mode: Host and Device A-Device Timeout Change (ADevTOUTChg) The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No A-Device Timeout</li> <li>■ 0x1 (ACTIVE): A-Device Timeout</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
17	HstNegDet	R/W1C	<p>Mode:Host and Device Host Negotiation Detected (HstNegDet) The core sets this bit when it detects a host negotiation request on the USB. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Active HNP Request</li> <li>■ 0x1 (ACTIVE): Active HNP request detected</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-7 Fields for Register: GOTGINT (Continued)**

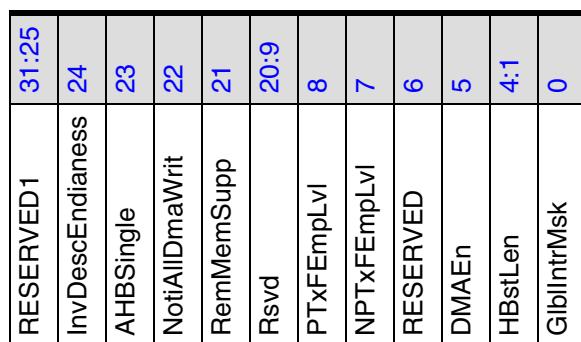
Bits	Name	Memory Access	Description
16:10	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
9	HstNegSucStsChng	R/W1C	<p>Mode: Host and Device</p> <p>Host Negotiation Success Status Change (HstNegSucStsChng)</p> <p>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. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Change</li> <li>■ 0x1 (ACTIVE): Host Negotiation Status Change</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
8	SesReqSucStsChng	R/W1C	<p>Mode: Host and Device</p> <p>Session Request Success Status Change (SesReqSucStsChng)</p> <p>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. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Change in Session Request Status</li> <li>■ 0x1 (ACTIVE): Session Request Status has changed</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
7:3	RESERVED2	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-7 Fields for Register: GOTGINT (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
2	SesEndDet	R/W1C	<p>Mode: Host and Device Session End Detected (SesEndDet) The controller sets this bit when the utmiotg_bvalid signal is deasserted. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Session is Active</li> <li>■ 0x1 (ACTIVE): SessionEnd utmiotg_bvalid signal is deasserted</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
1:0	RESERVED3	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead</p>

### 7.1.3 GAHBCFG

- **Name:** AHB Configuration Register
- **Description:** This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.
- **Size:** 32 bits
- **Offset:** 0x8
- **Exists:** Always



**Table 7-8 Fields for Register: GAHBCFG**

Bits	Name	Memory Access	Description
31:25	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-8 Fields for Register: GAHBCFG (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
24	InvDescEndianess	R/W	<p>Mode: Host and Device Invert Descriptor Endianess (InvDescEndianess)</p> <ul style="list-style-type: none"> <li>■ 1'b0: Descriptor Endianness is same as AHB Master Endianness.</li> <li>■ 1'b1: <ul style="list-style-type: none"> <li>- If the AHB Master endianness is Big Endian, the Descriptor Endianness is Little Endian.</li> <li>- If the AHB Master endianness is Little Endian, the Descriptor Endianness is Big Endian.</li> </ul> </li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Descriptor Endianness is same as AHB Master Endianness</li> <li>■ 0x1 (ENABLE): Descriptor Endianness is opposite to AHB Master Endianness</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA == 1</p>
23	AHBSingle	R/W	<p>Mode: Host and Device AHB Single Support (AHBSingle)</p> <p>This bit when programmed supports Single transfers for the remaining data in a transfer when the core is operating in DMA mode.</p> <ul style="list-style-type: none"> <li>■ 1'b0: The remaining data in the transfer is sent using INCR burst size.</li> <li>■ 1'b1: The remaining data in the transfer is sent using Single burst size.</li> </ul> <p><b>Note:</b> If this feature is enabled, the AHB RETRY and SPLIT transfers still have INCR burst type. Enable this feature when the AHB Slave connected to the core does not support INCR burst (and when Split, and Retry transactions are not being used in the bus).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INCRBURST): The remaining data in the transfer is sent using INCR burst size</li> <li>■ 0x1 (SINGLEBURST): The remaining data in the transfer is sent using Single burst size</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-8 Fields for Register: GAHBCFG (Continued)**

Bits	Name	Memory Access	Description
22	NotiAllDmaWrit	R/W	<p>Mode: Host and Device  Notify All DMA Write Transactions (NotiAllDmaWrit)  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.</p> <ul style="list-style-type: none"> <li>■ GAHBCFG.NotiAllDmaWrit = 1  The 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.</li> <li>■ GAHBCFG.NotiAllDmaWrit = 0  The 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.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (ALLTRANS): The 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</li> <li>■ 0x0 (LASTTRANS)</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-8 Fields for Register: GAHBCFG (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
21	RemMemSupp	R/W	<p>Mode: Host and Device Remote Memory Support (RemMemSupp) This bit is programmed to enable the functionality to wait for the system DMA Done Signal for the DMA Write Transfers.</p> <ul style="list-style-type: none"> <li>■ GAHBCFG.RemMemSupp=1 The int_dma_req output signal is asserted when the 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 the controller. 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.</li> <li>■ GAHBCFG.RemMemSupp=0 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 Core Boundary and it does not wait for the sys_dma_done signal to complete the DATA transfers.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Remote Memory Support Feature disabled</li> <li>■ 0x1 (ENABLED): Remote Memory Support Feature enabled</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
20:9			<b>Reserved Field:</b> Yes

**Table 7-8 Fields for Register: GAHBCFG (Continued)**

Bits	Name	Memory Access	Description
8	PTxFEmpLvl	R/W	<p>Mode: Host only  Periodic TxFIFO Empty Level (PTxFEmpLvl)  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> <ul style="list-style-type: none"> <li>■ 1'b0: GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is half empty</li> <li>■ 1'b1: GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is completely empty</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (HALFEMPTY): GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is half empty.</li> <li>■ 0x1 (EMPTY): GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is completely empty.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 3 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_EN_PERIO_HOST !=0</p>

**Table 7-8 Fields for Register: GAHBCFG (Continued)**

Bits	Name	Memory Access	Description
7	NPTxFEmpLvl	R/W	<p>Mode: Host and device Non-Periodic TxFIFO Empty Level (NPTxFEmpLvl) 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><b>Host mode and with Shared FIFO with device mode:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: GINTSTS.NPTxFEmp interrupt indicates that the Non-Periodic TxFIFO is half empty</li> <li>■ 1'b1: GINTSTS.NPTxFEmp interrupt indicates that the Non-Periodic TxFIFO is completely empty</li> </ul> <p><b>Dedicated FIFO in device mode:</b></p> <ul style="list-style-type: none"> <li>■ 1'b0: DIEPINTn.TxFEmp interrupt indicates that the IN Endpoint TxFIFO is half empty</li> <li>■ 1'b1: DIEPINTn.TxFEmp interrupt indicates that the IN Endpoint TxFIFO is completely empty</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (HALFEMPTY): DIEPINTn.TxFEmp interrupt indicates that the Non-Periodic TxFIFO is half empty or that the IN Endpoint TxFIFO is half empty.</li> <li>■ 0x1 (EMPTY): GINTSTS.NPTxFEmp interrupt indicates that the Non-Periodic TxFIFO is completely empty or that the IN Endpoint TxFIFO is completely empty.</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
6	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-8 Fields for Register: GAHBCFG (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
5	DMAEn	R/W	<p>Mode: Host and device DMA Enable (DMAEn)  This bit is always 0 when Slave-Only mode has been selected.</p> <p><b>Reset:</b> 1'b0</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (SLAVEMODE): Core operates in Slave mode</li> <li>■ 0x1 (DMAMODE): Core operates in a DMA mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE != 0</p>

**Table 7-8 Fields for Register: GAHBCFG (Continued)**

Bits	Name	Memory Access	Description
4:1	HBstLen	OTG_AR CHTECT URE == 0 ? R:R/W	<p>Mode: Host and device Burst Length/Type (HBstLen)</p> <p>This field is used in both External and Internal DMA modes. In External DMA mode, these bits appear on dma_burst[3:0] ports, which can be used by an external wrapper to interface the External DMA Controller interface to Synopsys DW_ahb_dmac or ARM PrimeCell.</p> <p>External DMA Mode defines the DMA burst length in terms of 32-bit words:</p> <ul style="list-style-type: none"> <li>■ 4'b0000: 1 word</li> <li>■ 4'b0001: 4 words</li> <li>■ 4'b0010: 8 words</li> <li>■ 4'b0011: 16 words</li> <li>■ 4'b0100: 32 words</li> <li>■ 4'b0101: 64 words</li> <li>■ 4'b0110: 128 words</li> <li>■ 4'b0111: 256 words</li> <li>■ Others: Reserved</li> </ul> <p>Internal DMA Mode AHB Master burst type:</p> <ul style="list-style-type: none"> <li>■ 4'b0000 Single</li> <li>■ 4'b0001 INCR</li> <li>■ 4'b0011 INCR4</li> <li>■ 4'b0101 INCR8</li> <li>■ 4'b0111 INCR16</li> <li>■ Others: Reserved</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (WORD1ORSINGLE): 1 word or single</li> <li>■ 0x1 (WORD4ORINCR): 4 words or INCR</li> <li>■ 0x2 (WORD8): 8 words</li> <li>■ 0x3 (WORD16ORINCR4): 16 words or INCR4</li> <li>■ 0x4 (WORD32): 32 words</li> <li>■ 0x5 (WORD64ORINCR8): 64 words or INCR8</li> <li>■ 0x6 (WORD128): 128 words</li> <li>■ 0x7 (WORD256ORINCR16): 256 words or INCR16</li> <li>■ 0x8 (WORDX): Others reserved</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-8 Fields for Register: GAHBCFG (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
0	GlbIntrMsk	R/W	<p>Mode: Host and device Global Interrupt Mask (GlbIntrMsk) 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 controller.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask the interrupt assertion to the application</li> <li>■ 0x1 (NOMASK): Unmask the interrupt assertion to the application.</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

### 7.1.4 GUSBCFG

- **Name:** USB Configuration Register
- **Description:** This register can be used to configure the core after power-on or when changing to Host mode or Device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. If you are using the HSIC interface, HSIC PHY must be in reset while programming this register. Do not make changes to this register after the initial programming.
- **Size:** 32 bits
- **Offset:** 0xc
- **Exists:** Always

CorruptTxPkt	31	ForceDevMode	30	ForceHstMode	29	TxEndDelay	28	IC_USBTrafCtl	27	IC_USBCap	26	ULPI	25	Indicator	24	Complement	23	TermSelDLPPulse	22	ULPIExtVbusIndic	21	ULPIExtVbusDrv	20	ULPIClkSusM	19	ULPIAutoRes	18	ULPIFsls	17	OrgI2CSel	16	PhyLPwrClikSel	15	Reserved_14	14	USBTrdTim	13:10	HNPCap	9	SRPCap	8	DDRSel	7	PHYSel	6	FSIntf	5	ULPI_UTMI_Sel	4	PHYIf	3	TOutCal	2:0
--------------	----	--------------	----	--------------	----	------------	----	---------------	----	-----------	----	------	----	-----------	----	------------	----	-----------------	----	------------------	----	----------------	----	-------------	----	-------------	----	----------	----	-----------	----	----------------	----	-------------	----	-----------	-------	--------	---	--------	---	--------	---	--------	---	--------	---	---------------	---	-------	---	---------	-----

Table 7-9 Fields for Register: GUSBCFG

Bits	Name	Memory Access	Description
31	CorruptTxPkt	W	<p>Mode: Host and device Corrupt Tx packet (CorruptTxPkt) This bit is for debug purposes only. Never Set this bit to 1. The application should always write 1'b0 to this bit.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NODEBUG): Normal Mode</li> <li>■ 0x1 (DEBUG): Debug Mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

Bits	Name	Memory Access	Description
30	ForceDevMode	R/W	<p>Mode:Host and device Force Device Mode (ForceDevMode) Writing a 1 to this bit forces the controller to device mode irrespective of utmiotg_iddig input pin.</p> <ul style="list-style-type: none"> <li>■ 1'b0 : Normal Mode.</li> <li>■ 1'b1 : Force Device Mode.</li> </ul> <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 micro sec is sufficient. This bit is valid only when OTG_MODE = 0, 1 or 2. In all other cases, this bit reads 0.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Normal Mode</li> <li>■ 0x1 (ENABLED): Force Device Mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 3 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 5 &amp;&amp; OTG_MODE != 6</p> <p><b>Testable:</b> untestable</p>
29	ForceHstMode	R/W	<p>Mode: Host and device Force Host Mode (ForceHstMode) Writing a 1 to this bit forces the core to host mode irrespective of utmiotg_iddig input pin.</p> <ul style="list-style-type: none"> <li>■ 1'b0 : Normal Mode.</li> <li>■ 1'b1 : Force Host Mode.</li> </ul> <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 micro sec is sufficient. This bit is valid only when OTG_MODE = 0, 1 or 2. In all other cases, this bit reads 0.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Normal Mode</li> <li>■ 0x1 (ENABLED): Force Host Mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 3 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 5 &amp;&amp; OTG_MODE != 6</p> <p><b>Testable:</b> untestable</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
28	TxEndDelay	R/W	<p>Mode: Device only Tx End Delay (TxEndDelay) Writing 1'b1 to this bit enables the controller to follow the TxEndDelay timings as per UTMI+ specification 1.05 section 4.1.5 for opmode signal during remote wakeup.</p> <ul style="list-style-type: none"> <li>■ 1'b0 : Normal Mode.</li> <li>■ 1'b1 : Tx End delay.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Normal Mode</li> <li>■ 0x1 (ENABLED): Tx End delay</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE == 0    OTG_MODE == 1    OTG_MODE == 2    OTG_MODE == 3    OTG_MODE == 4</p>
27	IC_USB_TrafficCtl	R/W	<p>Mode: Device only IC_USB-TrafficPullRemove Control When this bit is set, pullup/pulldown resistors are detached from the USB during traffic signaling, according to section 6.3.4 of the IC_USB specification. This bit is valid only when configuration parameter OTG_ENABLE_IC_USB = 1 and register field GUSBCFG.IC_USBCap is set to 1.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Pullup/pulldown resistors are not detached from the USB during traffic signaling</li> <li>■ 0x1 (ENABLED): Pullup/pulldown resistors are detached from the USB during traffic signaling</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6) &amp;&amp; OTG_ENABLE_IC_USB == 1</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

Bits	Name	Memory Access	Description
26	IC_USBCap	{OTG_EN ABLE_IC _USB == 1 & OTG_FS PHY_INT ERFACE != 0 ? R/W : R}	<p>Mode: Host and Device IC_USB-Capable (IC_USBCap) The application uses this bit to control the core's IC_USB capabilities.</p> <ul style="list-style-type: none"> <li>■ 1'b0: IC_USB PHY Interface is not selected.</li> <li>■ 1'b1: IC_USB PHY Interface is selected.</li> </ul> <p>This bit is writable only if OTG_ENABLE_IC_USB=1 and OTG_FSPHY_INTERFACE!=0.</p> <p>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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTSELECTED): IC_USB PHY Interface is not selected</li> <li>■ 0x1 (SELECTED): IC_USB PHY Interface is selected</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
25	ULPI	R/W	<p>Mode: Host only 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. For more information, refer to the ULPI Specification.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Enables the interface protect circuit</li> <li>■ 1'b1: Disables the interface protect circuit</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ENABLED): Enables the interface protect circuit</li> <li>■ 0x1 (DISABLED): Disables the interface protect circuit</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE !=3 &amp;&amp; OTG_MODE !=4) &amp;&amp; (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

Bits	Name	Memory Access	Description
24	Indicator	R/W	<p>Mode: Host only 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. For more information, refer to the ULPI Specification.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Complement Output signal is qualified with the Internal VbusValid comparator.</li> <li>■ 1'b1: Complement Output signal is not qualified with the Internal VbusValid comparator.</li> </ul> <p>This bit is reserved and read-only when OTG_HSPHY_INTERFACE is set to 0 or 1.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (QUALIFIED): Complement Output signal is qualified with the Internal VbusValid comparator</li> <li>■ 0x1 (NOTQUALIFIED): Complement Output signal is not qualified with the Internal VbusValid comparator</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE !=3 &amp;&amp; OTG_MODE !=4) &amp;&amp; (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)</p>
23	Complement	R/W	<p>Mode: Host only Indicator Complement Controls the PHY to invert the ExternalVbusIndicator input signal, generating the Complement Output. For more information, refer to the ULPI Specification.</p> <ul style="list-style-type: none"> <li>■ 1'b0: PHY does not invert ExternalVbusIndicator signal</li> <li>■ 1'b1: PHY does invert ExternalVbusIndicator signal</li> </ul> <p>This bit is reserved and read-only when OTG_HSPHY_INTERFACE is set to 0 or 1.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NONINVERT): PHY does not invert ExternalVbusIndicator signal</li> <li>■ 0x1 (INVERT): PHY inverts ExternalVbusIndicator signal</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE !=3 &amp;&amp; OTG_MODE !=4) &amp;&amp; (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

Bits	Name	Memory Access	Description
22	TermSelDLPulse	OTG_HS PHY_INT ERFACE!=0 ? R/W : R	<p>Mode: Device only</p> <p>TermSel DLine Pulsing Selection (TermSelDLPulse)</p> <p>This bit selects utmi_termselect to drive data line pulse during SRP.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Data line pulsing using utmi_txvalid (Default).</li> <li>■ 1'b1: Data line pulsing using utmi_termsel.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (TXVALID): Data line pulsing using utmi_txvalid</li> <li>■ 0x1 (TERMSEL): Data line pulsing using utmi_termsel</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6</p>
21	ULPIExtVbusIndicator	R/W	<p>Mode: Host only</p> <p>ULPI External VBUS Indicator (ULPIExtVbusIndicator)</p> <p>This bit indicates to the ULPI PHY to use an external VBUS overcurrent indicator.</p> <ul style="list-style-type: none"> <li>■ 1'b0: PHY uses internal VBUS valid comparator.</li> <li>■ 1'b1: PHY uses external VBUS valid comparator.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INTERN): PHY uses internal VBUS valid comparator</li> <li>■ 0x1 (EXTERN): PHY uses external VBUS valid comparator</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4) &amp;&amp; (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)</p>
20	ULPIExtVbusDrv	R/W	<p>Mode: Host only</p> <p>ULPI External VBUS Drive (ULPIExtVbusDrv)</p> <p>This bit selects between internal or external supply to drive 5V on VBUS, in ULPI PHY.</p> <ul style="list-style-type: none"> <li>■ 1'b0: PHY drives VBUS using internal charge pump (Default).</li> <li>■ 1'b1: PHY drives VBUS using external supply.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INTERN): PHY drives VBUS using internal charge pump</li> <li>■ 0x1 (EXTERN): PHY drives VBUS using external supply</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4) &amp;&amp; (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

Bits	Name	Memory Access	Description
19	ULPIClkSusM	R/W	<p>Mode: Host and Device ULPI Clock SuspendM (ULPIClkSusM) This bit sets the ClockSuspendM bit in the Interface Control register on the ULPI PHY. This bit applies only in serial or carkit modes.</p> <ul style="list-style-type: none"> <li>■ 1'b0: PHY powers down internal clock during suspend.</li> <li>■ 1'b1: PHY does not power down internal clock.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (PWDCLK): PHY powers down internal clock during suspend</li> <li>■ 0x1 (NONPWDCLK): PHY does not power down internal clock</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)</p>
18	ULPIAutoRes	R/W	<p>Mode: Host and Device ULPI Auto Resume (ULPIAutoRes) This bit sets the AutoResume bit in the Interface Control register on the ULPI PHY.</p> <ul style="list-style-type: none"> <li>■ 1'b0: PHY does not use AutoResume feature.</li> <li>■ 1'b1: PHY uses AutoResume feature.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): PHY does not use AutoResume feature</li> <li>■ 0x1 (ENABLED): PHY uses AutoResume feature</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

Bits	Name	Memory Access	Description
17	ULPIFsLs	R/W	<p>Mode: Host and Device ULPI FS/LS Select (ULPIFsLs) The application uses this bit to select the FS/LS serial interface for the ULPI PHY. This bit is valid only when the FS serial transceiver is selected on the ULPI PHY.</p> <ul style="list-style-type: none"> <li>■ 1'b0: ULPI interface</li> <li>■ 1'b1: ULPI FS/LS serial interface</li> </ul> <p><b>Note:</b> Before setting this bit, the application needs to ensure that GUSBCFG.ULPI_UTMI_SEL = 1'b1.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): ULPI interface is selected</li> <li>■ 0x1 (ENABLED): ULPI FS/LS serial interface is selected</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> ((OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3) &amp;&amp; (OTG_FSPHY_INTERFACE == 1    OTG_FSPHY_INTERFACE == 2    OTG_FSPHY_INTERFACE == 3))</p> <p><b>Testable:</b> restore</p>
16	Otgl2CSel	OTG_I2C_INTERFACE == 2 ? R/W : R	<p>Mode: Host and Device UTMIFS or I2C Interface Select (Otgl2CSel) The application uses this bit to select the I2C interface.</p> <ul style="list-style-type: none"> <li>■ 1'b0: UTMI USB 1.1 Full-Speed interface for OTG signals</li> <li>■ 1'b1: I2C interface for OTG signals</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): UTMI USB 1.1 Full-Speed interface for OTG signals</li> <li>■ 0x1 (ENABLED): I2C interface for OTG signals</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_I2C_INTERFACE == 2)</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

Bits	Name	Memory Access	Description
15	PhyPwrClkSel	R/W	<p>PHY Low-Power Clock Select (PhyPwrClkSel)  <b>Mode:</b> Host and Device  Selects either 480-MHz or 48-MHz (low-power) PHY mode. In FS and LS modes, the PHY can usually operate on a 48-MHz clock to save power.</p> <ul style="list-style-type: none"> <li>■ 1'b0: 480-MHz Internal PLL clock</li> <li>■ 1'b1: 48-MHz External Clock</li> </ul> <p>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 mode and at either 48 or 6 MHz in LS mode (depending on the PHY vendor). This bit drives the <code>utmi_fs_ls_low_power</code> core output signal, and is valid only for UTMI+ PHYs.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INTPLLCLK): 480-MHz Internal PLL clock</li> <li>■ 0x1 (EXTCLK): 48-MHz External Clock</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_HSPHY_INTERFACE != 0 &amp;&amp; OTG_HSPHY_INTERFACE != 2</p>
14	Reserved_14	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

Bits	Name	Memory Access	Description
13:10	USBTrdTim	R/W	<p>Mode: Device only          USB Turnaround Time (USBTrdTim)          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 DFIFO (SPRAM). This must be programmed to</p> <ul style="list-style-type: none"> <li>■ 4'h5 : When the MAC interface is 16-bit UTMI+</li> <li>■ 4'h9 : When the MAC interface is 8-bit UTMI+</li> </ul> <p><b>Note:</b> Programming Guide Section "Choosing the Value of GUSBCFG.USBTrdTim" consider mis-sampling into account whereas above values are without mis-sampling. During the AHB Clock Frequency selection, it is recommended to consider mis-sampling into account for USBTrdTim. <b>Note:</b> The previous values 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. 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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x9 (TURNTIME8BIT): MAC interface is 8-bit UTMI+.</li> <li>■ 0x5 (TURNTIME16BIT): MAC interface is 16-bit UTMI+.</li> </ul> <p><b>Value After Reset:</b> 0x5</p> <p><b>Exists:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6</p>
9	HNPCap	OTG_MODE == 0 ? R/W : R	<p>Mode: Host and Device          HNP-Capable (HNPCap)          The application uses this bit to control the controller's HNP capabilities.</p> <ul style="list-style-type: none"> <li>■ 1'b0: HNP capability is not enabled.</li> <li>■ 1'b1: HNP capability is enabled.</li> </ul> <p>If HNP functionality is disabled by the software, the OTG signals on the PHY domain must be tied to the appropriate values.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): HNP capability is not enabled.</li> <li>■ 0x1 (ENABLED): HNP capability is enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 1 &amp;&amp; OTG_MODE != 2 &amp;&amp; OTG_MODE != 3 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 5 &amp;&amp; OTG_MODE != 6</p> <p><b>Testable:</b> restore</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
8	SRPCap	OTG_MO DE != 2 & OTG_MO DE != 4 ? R/W : R	<p>Mode: Host and Device SRP-Capable (SRPCap)</p> <p>The application uses this bit to control the controller's 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> <ul style="list-style-type: none"> <li>■ 1'b0: SRP capability is not enabled.</li> <li>■ 1'b1: SRP capability is enabled.</li> </ul> <p>If SRP functionality is disabled by the software, the OTG signals on the PHY domain must be tied to the appropriate values.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): SRP capability is not enabled</li> <li>■ 0x1 (ENABLED): SRP capability is enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 2 &amp;&amp; OTG_MODE != 4 &amp;&amp; OTG_MODE != 6</p> <p><b>Testable:</b> restore</p>
7	DDRSel	R/W	<p>Mode: Host and Device ULPI DDR Select (DDRSel)</p> <p>The application uses this bit to select a Single Data Rate (SDR) or Double Data Rate (DDR) or ULPI interface.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Single Data Rate ULPI Interface, with 8-bit-wide data bus</li> <li>■ 1'b1: Double Data Rate ULPI Interface, with 4-bit-wide data bus</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (SDR): Single Data Rate ULPI Interface with 8-bit-wide data bus</li> <li>■ 0x1 (DDR): Double Data Rate ULPI Interface with 4-bit-wide data bus</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_HSPHY_INTERFACE == 2    OTG_HSPHY_INTERFACE == 3)</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

Bits	Name	Memory Access	Description
6	PHYSel	OTG_HS PHY_INT ERFACE! =0 & OTG_FS PHY_INT ERFACE! =0 ? R/W : R	<p>PHYSel Mode: Host and Device USB 2.0 High-Speed PHY or USB 1.1 Full-Speed Serial Transceiver Select (PHYSel)</p> <p>The application uses this bit to select either a high-speed UTMI+ or ULPI PHY, or a full-speed transceiver.</p> <ul style="list-style-type: none"> <li>■ 1'b0: USB 2.0 high-speed UTMI+ or ULPI PHY</li> <li>■ 1'b1: USB 1.1 full-speed serial transceiver</li> </ul> <p>If a USB 1.1 Full-Speed Serial Transceiver interface was not selected in, this bit is always 0, with Write Only access.</p> <p>If a high-speed PHY interface was not selected in, this bit is always 1, with Write Only access.</p> <p>If both interface types were selected (parameters have non-zero values), the application uses this bit to select which interface is active, and access is Read and Write.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (USB20): USB 2.0 high-speed UTMI+ or ULPI PHY is selected</li> <li>■ 0x1 (USB11): USB 1.1 full-speed serial transceiver is selected</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
5	FSIntf	OTG_FS PHY_INT ERFACE! =0 ? R/W : R	<p>Mode: Host and Device Full-Speed Serial Interface Select (FSIntf)</p> <p>The application uses this bit to select either a unidirectional or bidirectional USB 1.1 full-speed serial transceiver interface.</p> <ul style="list-style-type: none"> <li>■ 1'b0: 6-pin unidirectional full-speed serial interface</li> <li>■ 1'b1: 3-pin bidirectional full-speed serial interface</li> </ul> <p>If a USB 1.1 Full-Speed Serial Transceiver interface was not selected, this bit is always 0, with Write Only access. If a USB 1.1 FS interface was selected, Then the application can Set this bit to select between the 3- and 6-pin interfaces, and access is Read and Write.</p> <p><b>Note:</b> For supporting the new 4-pin bi-directional interface, you need to select 6-pin unidirectional FS serial mode, and add an external control to convert it to a 4-pin interface.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (FS6PIN): 6-pin unidirectional full-speed serial interface</li> <li>■ 0x1 (FS3PIN): 3-pin bidirectional full-speed serial interface</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

Bits	Name	Memory Access	Description
4	ULPI_UTMI_Sel	(OTG_HS PHY_INT ERFACE) == 3) ? R/W : ((OTG_H SPHY_IN TERFAC E] == 2) & (OTG_FS PHY_INT ERFACE == 3))? R/W:R	<p>Mode: Host and Device ULPI or UTMI+ Select (ULPI_UTMI_Sel) The application uses this bit to select either a UTMI+ interface or ULPI Interface.</p> <ul style="list-style-type: none"> <li>■ 1'b0: UTMI+ Interface</li> <li>■ 1'b1: ULPI Interface</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (UTMI): UTMI+ Interface</li> <li>■ 0x1 (ULPI): ULPI Interface</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_HSPHY_INTERFACE != 0</p> <p><b>Volatile:</b> true</p>
3	PHYIf	OTG_HS PHY_DW IDTH==2 ? R/W : R	<p>Mode: Host and Device PHY Interface (PHYIf)</p> <p>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.</p> <ul style="list-style-type: none"> <li>■ 1'b0: 8 bits</li> <li>■ 1'b1: 16 bits</li> </ul> <p>This bit is writable only If UTMI+ and ULPI were selected. Otherwise, this bit returns the value for the power-on interface selected during configuration.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (BITS8): PHY 8bit Mode</li> <li>■ 0x1 (BITS16): PHY 16bit Mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-9 Fields for Register: GUSBCFG (Continued)**

Bits	Name	Memory Access	Description
2:0	TOutCal	R/W	<p>Mode: Host and Device HS/FS Timeout Calibration (TOutCal) 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 linestate condition can vary from one PHY to another.</p> <p>The USB standard timeout value for high-speed operation is 736 to 816 (inclusive) bit times. The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock are as follows:</p> <p><b>High-speed operation:</b></p> <ul style="list-style-type: none"> <li>■ One 30-MHz PHY clock = 16 bit times</li> <li>■ One 60-MHz PHY clock = 8 bit times</li> </ul> <p><b>Full-speed operation:</b></p> <ul style="list-style-type: none"> <li>■ One 30-MHz PHY clock = 0.4 bit times</li> <li>■ One 60-MHz PHY clock = 0.2 bit times</li> <li>■ One 48-MHz PHY clock = 0.25 bit times</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ZERO): Add 0 PHY clocks</li> <li>■ 0x1 (ONE): Add 1 PHY clocks</li> <li>■ 0x2 (TWO): Add 2 PHY clocks</li> <li>■ 0x3 (THREE): Add 3 PHY clocks</li> <li>■ 0x4 (FOUR): Add 4 PHY clocks</li> <li>■ 0x5 (FIVE): Add 5 PHY clocks</li> <li>■ 0x6 (SIX): Add 6 PHY clocks</li> <li>■ 0x7 (SEVEN): Add 7 PHY clocks</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.5 GRSTCTL

- **Name:** Reset Register
- **Description:** The application uses this register to reset various hardware features inside the controller.
- **Size:** 32 bits
- **Offset:** 0x10
- **Exists:** Always

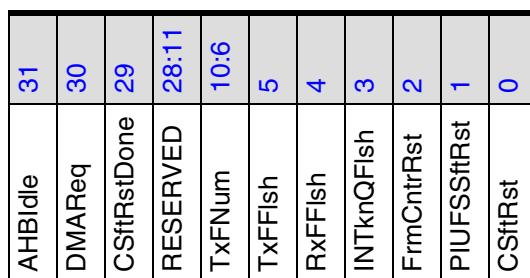


Table 7-10 Fields for Register: GRSTCTL

Bits	Name	Memory Access	Description
31	AHBIdle	R	<p>Mode: Host and Device AHB Master Idle (AHBIdle) Indicates that the AHB Master State Machine is in the IDLE condition.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not Idle</li> <li>■ 0x1 (ACTIVE): AHB Master Idle</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p>
30	DMAReq	R	<p>Mode: Host and Device DMA Request Signal (DMAReq) Indicates that the DMA request is in progress. Used for debug.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No DMA request</li> <li>■ 0x1 (ACTIVE): DMA request is in progress</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-10 Fields for Register: GRSTCTL (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
29	CSftRstDone	R/W1C	<p>Mode: Host and Device  Core Soft Reset Done (CSftRstDone)  The core sets this bit when all the necessary logic is reset in the core. This bit is cleared by the application along with GRSTCTL.CSftRst (bit 0)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No reset</li> <li>■ 0x1 (ACTIVE): Core Soft Reset is done</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
28:11	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-10 Fields for Register: GRSTCTL (Continued)**

Bits	Name	Memory Access	Description
10:6	TxFNum	R/W	<p>Mode: Host and Device TxFIFO Number (TxFNum) 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.</p> <ul style="list-style-type: none"> <li>■ 5'h0: <ul style="list-style-type: none"> <li>- Non-periodic TxFIFO flush in Host mode</li> <li>- Non-periodic TxFIFO flush in device mode when in shared FIFO operation</li> <li>- Tx FIFO 0 flush in device mode when in dedicated FIFO mode</li> </ul> </li> <li>■ 5'h1: <ul style="list-style-type: none"> <li>- Periodic TxFIFO flush in Host mode</li> <li>- Periodic TxFIFO 1 flush in Device mode when in shared FIFO operation</li> <li>- TXFIFO 1 flush in device mode when in dedicated FIFO mode</li> </ul> </li> <li>■ 5'h2: <ul style="list-style-type: none"> <li>- Periodic TxFIFO 2 flush in Device mode when in shared FIFO operation</li> <li>- TXFIFO 2 flush in device mode when in dedicated FIFO mode</li> </ul> </li> <li>...</li> <li>■ 5'hF <ul style="list-style-type: none"> <li>- Periodic TxFIFO 15 flush in Device mode when in shared FIFO operation</li> <li>- TXFIFO 15 flush in device mode when in dedicated FIFO mode</li> </ul> </li> <li>■ 5'h10: Flush all the transmit FIFOs in device or host mode</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (TXF0): -Periodic TxFIFO flush in host mode -Periodic TxFIFO 0 flush in device mode when in shared FIFO operation -TxFIFO 0 flush in device mode when in dedicated FIFO mode</li> <li>■ 0x1 (TXF1): -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</li> <li>■ 0x2 (TXF2): -Periodic TxFIFO 2 flush in device mode when in shared FIFO operation -TxFIFO 2 flush in device mode when in dedicated FIFO mode</li> </ul>

**Table 7-10 Fields for Register: GRSTCTL (Continued)**

Bits	Name	Memory Access	Description
			<ul style="list-style-type: none"> <li>■ 0x3 (TXF3): -Periodic TxFIFO 3 flush in device mode when in shared FIFO operation -TxFIFO 3 flush in device mode when in dedicated FIFO mode</li> <li>■ 0x4 (TXF4): -Periodic TxFIFO 4 flush in device mode when in shared FIFO operation -TxFIFO 4 flush in device mode when in dedicated FIFO mode</li> <li>■ 0x5 (TXF5): -Periodic TxFIFO 5 flush in device mode when in shared FIFO operation -TxFIFO 5 flush in device mode when in dedicated FIFO mode</li> <li>■ 0x6 (TXF6): -Periodic TxFIFO 6 flush in device mode when in shared FIFO operation -TxFIFO 6 flush in device mode when in dedicated FIFO mode</li> <li>■ 0x7 (TXF7): -Periodic TxFIFO 7 flush in device mode when in shared FIFO operation -TxFIFO 7 flush in device mode when in dedicated FIFO mode</li> <li>■ 0x8 (TXF8): -Periodic TxFIFO 8 flush in device mode when in shared FIFO operation -TxFIFO 8 flush in device mode when in dedicated FIFO mode</li> <li>■ 0x9 (TXF9): -Periodic TxFIFO 9 flush in device mode when in shared FIFO operation -TxFIFO 9 flush in device mode when in dedicated FIFO mode</li> <li>■ 0xa (TXF10): -Periodic TxFIFO 10 flush in device mode when in shared FIFO operation -TxFIFO 10 flush in device mode when in dedicated FIFO mode</li> <li>■ 0xb (TXF11): -Periodic TxFIFO 11 flush in device mode when in shared FIFO operation -TxFIFO 11 flush in device mode when in dedicated FIFO mode</li> <li>■ 0xc (TXF12): -Periodic TxFIFO 12 flush in device mode when in shared FIFO operation -TxFIFO 12 flush in device mode when in dedicated FIFO mode</li> <li>■ 0xd (TXF13): -Periodic TxFIFO 13 flush in Device mode when in shared FIFO operation -TxFIFO 13 flush in device mode when in dedicated FIFO mode</li> <li>■ 0xe (TXF14): -Periodic TxFIFO 14 flush in Device mode when in shared FIFO operation -TxFIFO 14 flush in device mode when in dedicated FIFO mode</li> <li>■ 0xf (TXF15): -Periodic TxFIFO 15 flush in Device mode when in shared FIFO operation -TxFIFO 15 flush in device mode when in dedicated FIFO mode</li> <li>■ 0x10 (TXF16): Flush all the transmit FIFOs in device or host mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_EN_PERIO_HOST == 1)    (OTG_NUM_PERIO_EPS != 0)    (OTG_EN_DED_TX_FIFO == 1)</p>

**Table 7-10 Fields for Register: GRSTCTL (Continued)**

Bits	Name	Memory Access	Description
5	TxFFlsh	R/W1S	<p>Mode: Host and Device TxFIFO Flush (TxFFlsh) 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:</p> <ul style="list-style-type: none"> <li>■ ReadNAK Effective Interrupt ensures the core is not reading from the FIFO</li> <li>■ WriteGRSTCTL.AHBIdle ensures the core is not writing anything to the FIFO.</li> </ul> <p>Flushing is normally recommended when FIFOs are reconfigured or when switching between Shared FIFO and Dedicated Transmit FIFO operation. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Flush</li> <li>■ 0x1 (ACTIVE): Selectively flushes a single or all transmit FIFOs</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>
4	RxFFlsh	R/W1S	<p>Mode: Host and Device RxFIFO Flush (RxFFlsh) The application can flush the entire RxFIFO 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 controller is neither reading from the RxFIFO nor writing to the RxFIFO. The application must wait until the bit is cleared before performing any other operations. This bit requires eight clocks (slowest of PHY or AHB clock) to clear.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Does not flush the entire RxFIFO</li> <li>■ 0x1 (ACTIVE): Flushes the entire RxFIFO</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>

**Table 7-10 Fields for Register: GRSTCTL (Continued)**

Bits	Name	Memory Access	Description
3	INTknQFlsh	R/W1S	<p>Mode: Device only  IN Token Sequence Learning Queue Flush (INTknQFlsh)  The application writes this bit to flush the IN Token Sequence Learning Queue.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Flush</li> <li>■ 0x1 (ACTIVE): IN Token Sequence Learning Queue Flush</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_ENDED_TX_FIFO == 0)</p> <p><b>Testable:</b> readOnly</p>
2	FrmCntrRst	R/W1S	<p>Mode: Host only  Host Frame Counter Reset (FrmCntrRst)  The application writes this bit to reset the (micro)Frame number counter inside the core. When the (micro)Frame counter is reset, the subsequent SOF sent out by the core has a (micro)Frame number of 0.</p> <p>When application writes 1 to the bit, it might not be able to read back the value as it will get cleared by the core in a few clock cycles.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTACTIVE): No reset</li> <li>■ 0x1 (ACTIVE): Host Frame Counter Reset</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4)</p> <p><b>Testable:</b> readOnly</p>

**Table 7-10 Fields for Register: GRSTCTL (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
1	PIUFSSftRst	R/W1S	<p>Mode: Host and Device      PIU FS Dedicated Controller Soft Reset (PIUFSSftRst)      Resets the PIU FS Dedicated Controller      All module state machines in FS Dedicated Controller of PIU are reset to the IDLE state. Used to reset the FS Dedicated controller in PIU in case of any PHY Errors like Loss of activity or Babble Error resulting in the PHY remaining in RX state for more than one frame boundary.      This is a self clearing bit and core clears this bit after all the necessary logic is reset in the core.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (RESET_INACTIVE): No Reset</li> <li>■ 0x1 (RESET_ACTIVE): PIU FS Dedicated Controller Soft Reset</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always  <b>Testable:</b> writeAsRead</p>

**Table 7-10 Fields for Register: GRSTCTL (Continued)**

Bits	Name	Memory Access	Description
0	CSftRst	R/W	<p>Mode: Host and Device Core Soft Reset (CSftRst) Resets the hclk and phy_clock domains as follows:</p> <ul style="list-style-type: none"> <li>■ Clears the interrupts and all the CSR registers except the following register bits: <ul style="list-style-type: none"> <li>- PCGCCTL.RstPdwnModule</li> <li>- PCGCCTL.GateHclk</li> <li>- PCGCCTL.PwrClmp</li> <li>- PCGCCTL.StopPPhyLPwrClkSelClk</li> <li>- GUSBCFG.ForceDevMode</li> <li>- GUSBCFG.ForceHstMode</li> <li>- GUSBCFG.PhylPwrClkSel</li> <li>- GUSBCFG.DDRSel</li> <li>- GUSBCFG.PHYSel</li> <li>- GUSBCFG.FSIntf</li> <li>- GUSBCFG.ULPI_UTMI_Sel</li> <li>- GUSBCFG.PHYIf</li> <li>- GUSBCFG.TxEndDelay</li> <li>- GUSBCFG.TermSelDLPulse</li> <li>- GUSBCFG.ULPIClkSusM</li> <li>- GUSBCFG.ULPIAutoRes</li> <li>- GUSBCFG.ULPIFsLs</li> <li>- GPIO</li> <li>- GPWRDN</li> <li>- GADPCTL</li> <li>- HCFG.FSLSPclkSel</li> <li>- DCFG.DevSpd</li> <li>- DCTL.SftDiscon</li> </ul> </li> <li>■ All module state machines</li> <li>■ 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.</li> <li>■ 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.</li> <li>■ When Hibernation or ADP feature is enabled, the PMU module is not reset by the Core Soft Reset.</li> </ul>

**Table 7-10 Fields for Register: GRSTCTL (Continued)**

Bits	Name	Memory Access	Description
			<p>The application can write to this bit any time it wants to reset the core. The application must clear this bit after checking the bit 29 of this register (Core Soft Reset Done). Software must also check that bit 31 of this register is 1 (AHB Master is IDLE) before starting any operation.</p> <p>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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTACTIVE): No reset</li> <li>■ 0x1 (ACTIVE): Resets hclk and phy_clock domains</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>

## 7.1.6 GINTSTS

- **Name:** Interrupt Register
  - **Description:** This register interrupts the application for system-level events in the current mode (Device mode or Host mode).

Some of the bits in this register are valid only in Host mode, while others are valid in Device mode only. This register also indicates the current mode. To clear the interrupt status bits of type R\_SS\_WC, the application must write 1'b1 to the bit.

The FIFO status interrupts are read only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

**Note:** Read the reset value of GINTSTS.CurMod only after the following conditions:

- ❑ If IDDIG\_FILTER is disabled, read only after PHY clock is stable.
  - ❑ If IDDIG\_FILTER is enabled, read only after the filter timer expires.

- **Size:** 32 bits
  - **Offset:** 0x14
  - **Exists:** Always

**Table 7-11 Fields for Register: GINTSTS**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
31	WkUplnt	R/W1C	<p>Mode: Host and Device Resume/Remote Wakeup Detected Interrupt (WkUplnt) Wakeup Interrupt during Suspend(L2) or LPM(L1) state.</p> <ul style="list-style-type: none"> <li>■ During Suspend(L2): <ul style="list-style-type: none"> <li>- Device Mode: This interrupt is asserted only when Host Initiated Resume is detected on USB.</li> <li>- Host Mode: This interrupt is asserted only when Device Initiated Remote Wakeup is detected on USB.</li> </ul> </li> </ul> <p>For more information, see 'Partial Power-Down and Clock Gating Programming Model' in the Programming Guide.</p> <ul style="list-style-type: none"> <li>■ During LPM(L1): <ul style="list-style-type: none"> <li>- Device Mode: This interrupt is asserted for either Host Initiated Resume or Device Initiated Remote Wakeup on USB.</li> <li>- Host Mode: This interrupt is asserted for either Host Initiated Resume or Device Initiated Remote Wakeup on USB.</li> </ul> </li> </ul> <p>For more information, see 'LPM Entry and Exit Programming Model' in the Programming Guide.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Resume or Remote Wakeup Detected Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
30	SessReqInt	R/W1C	<p>Mode: Host and Device Session Request/New Session Detected Interrupt (SessReqInt) 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> <p>For more information on how to use this interrupt, see 'Partial Power-Down and Clock Gating Programming Model' in the Programming Guide.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Session Request New Session Detected Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
29	DisconnectInt	R/W1C	<p>Mode: Host only Disconnect Detected Interrupt (DisconnectInt) Asserted when a device disconnect is detected.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Disconnect Detected Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4 )</p>
28	ConIDStsChng	R/W1C	<p>Mode: Host and Device Connector ID Status Change (ConIDStsChng) The core sets this bit when there is a change in connector ID status.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not Active</li> <li>■ 0x1 (ACTIVE): Connector ID Status Change</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>
27	LPM_Int	R/W1C	<p>Mode: Host and Device LPM Transaction Received Interrupt (LPM_Int).</p> <ul style="list-style-type: none"> <li>■ Device Mode: This interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.</li> <li>■ 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).</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not Active</li> <li>■ 0x1 (ACTIVE): LPM Transaction Received Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6) &amp;&amp; (OTG_ENABLE_LPM==1)</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

Bits	Name	Memory Access	Description
26	PTxFEmp	R	<p>Mode: Host only Periodic TxFIFO Empty (PTxFEmp)</p> <p>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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Periodic TxFIFO Empty</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4) &amp;&amp; OTG_EN_PERIO_HOST ==1</p> <p><b>Volatile:</b> true</p>
25	HChInt	R	<p>Mode: Host only Host Channels Interrupt (HChInt)</p> <p>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.</p> <p>The application must clear the appropriate status bit in the HCINTn register to clear this bit.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Host Channels Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4 )</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
24	PrtInt	R	<p>Mode: Host only Host Port Interrupt (PrtInt) The core sets this bit to indicate a change in port status of one of the controller 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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Host Port Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4 )</p>
23	ResetDet	R/W1C	<p>Mode: Device only Reset detected Interrupt (ResetDet) In Device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in Suspend. In Host mode, this interrupt is not asserted.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Reset detected Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

Bits	Name	Memory Access	Description
22	FetSusp	R/W1C	<p>Mode: Device only Data Fetch Suspended (FetSusp) 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.</p> <p>For example, after detecting an endpoint mismatch, the application:</p> <ul style="list-style-type: none"> <li>■ Sets a Global non-periodic IN NAK handshake</li> <li>■ Disables IN endpoints</li> <li>■ Flushes the FIFO</li> <li>■ Determines the token sequence from the IN Token Sequence Learning Queue</li> <li>■ Re-enables the endpoints</li> <li>■ Clears the Global non-periodic IN NAK handshake</li> </ul> <p>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. It 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.</p> <p>Alternatively, the application can mask the IN token received when FIFO empty interrupt when clearing a Global IN NAK handshake.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Data Fetch Suspended</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

Bits	Name	Memory Access	Description
21	incomplP	R/W1C	<p><b>Incomplete Periodic Transfer (incomplP)</b>  Mode: Host only  In Host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending which are scheduled for the current microframe.</p> <p><b>Incomplete Isochronous OUT Transfer (incomplSOOUT)</b>  Mode: Device only  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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Incomplete Periodic Transfer</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
20	incompISOIN	R/W1C	<p>Mode: Device only  Incomplete Isochronous IN Transfer (incompISOIN)  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.</p> <p><b>Note:</b> This interrupt is not asserted in Scatter/Gather DMA mode.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Incomplete Isochronous IN Transfer</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
19	OEPInt	R	<p>Mode: Device only OUT Endpoints Interrupt (OEPInt) The controller 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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): OUT Endpoints Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>
18	IEPInt	R	<p>Mode: Device only IN Endpoints Interrupt (IEPInt) 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 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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): IN Endpoints Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

Bits	Name	Memory Access	Description
17	EPMis	R/W1C	<p>Mode: Device only Endpoint Mismatch Interrupt (EPMis) <b>Note:</b> This interrupt is valid only in shared FIFO operation. Indicates that an IN token has been received for a non-periodic endpoint, but the data for another endpoint is present in the top of the Non-periodic Transmit FIFO and the IN endpoint mismatch count programmed by the application has expired.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Endpoint Mismatch Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>
16	RstrDoneInt	R/W1C	<p>Mode: Device only Restore Done Interrupt (RstrDoneInt) The controller sets this bit to indicate that the restore command after Hibernation was completed by the core. The controller continues from Suspended state into the mode dictated by PCGCCTL.RestoreMode field.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Restore Done Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> (OTG_EN_PWROPT==2)</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

Bits	Name	Memory Access	Description
15	EOPF	R/W1C	<p>Mode: Device only End of Periodic Frame Interrupt (EOPF) Indicates that the period specified in the Periodic Frame Interval field of the Device Configuration register (DCFG.PerFrlnt) has been reached in the current microframe. In case of Non-Ignore Frame Number Scatter/Gather (Descriptor DMA) mode, the controller internally handles the following scenarios based on EOPF:</p> <p><b>Read Flush:</b> At the EOPF, the controller checks if there are any pending packets in the FIFO corresponding to the current (micro)Frame.</p> <ul style="list-style-type: none"> <li>■ If there are any pending packets, then the controller initiates read flush, due to which the read pointer is updated to the starting location of the next micro-frame packet.</li> <li>■ If there are no pending packets corresponding to the current (micro)Frame, the controller does not take any action.</li> </ul> <p><b>Write Flush:</b> At the EOPF, if the controller is still fetching the current micro-frame data, then the controller stops pushing data into the TXFIFO but keeps fetching the complete packet from the System Memory. After completing the scheduled packet size fetch, the controller updates the Status Quadlet Fields (Transmit Status to BUFFFLUSH) and closes the Descriptor. During the descriptor close, the controller initiates write flush, due to which the write pointer is updated to the starting location of the next micro-frame packet. Because the controller stops pushing the packet to the TxFIFO after EOPF, to bring the write pointer to the starting location of the next micro-frame, write flush is done.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): End of Periodic Frame Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>
14	ISOOutDrop	R/W1C	<p>Mode: Device only Isochronous OUT Packet Dropped Interrupt (ISOOutDrop) The controller sets this bit when it fails to write an isochronous OUT packet into the RxFIFO because the RxFIFO does not have enough space to accommodate a maximum packet size packet for the isochronous OUT endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Isochronous OUT Packet Dropped Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
13	EnumDone	R/W1C	<p>Mode: Device only Enumeration Done (EnumDone) 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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Enumeration Done</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>
12	USBRst	R/W1C	<p>Mode: Device only USB Reset (USBRst) The controller sets this bit to indicate that a reset is detected on the USB.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): USB Reset</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>
11	USBSusp	R/W1C	<p>Mode: Device only USB Suspend (USBSusp) The controller sets this bit to indicate that a suspend was detected on the USB. The controller enters the Suspended state when there is no activity on the linestate signal for an extended period of time.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not Active</li> <li>■ 0x1 (ACTIVE): USB Suspend</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>
10	ErlySusp	R/W1C	<p>Mode: Device only Early Suspend (ErlySusp) The controller sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Idle state detected</li> <li>■ 0x1 (ACTIVE): 3ms of Idle state detected</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

Bits	Name	Memory Access	Description
9	I2CINT	R/W1C	<p>Mode: Host and Device I2C Interrupt (I2CINT) The controller sets this interrupt when I2C access is completed on the I2C interface.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not Active</li> <li>■ 0x1 (ACTIVE): I2C Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_I2C_INTERFACE == 1)</p>
8	ULPICKINT	R/W1C	<p>Mode: Host and Device ULPI Carkit Interrupt (ULPICKINT) 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.</p> <p>I2C Carkit Interrupt (I2CCKINT) The controller sets this interrupt when a Carkit interrupt is received. The PHY sets the I2C Carkit interrupt in Audio mode.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not Active</li> <li>■ 0x1 (ACTIVE): Interrupt is asserted high when ULPI Carkit Interrupt is received in case of Carkit or I2C interfaces</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_I2C_INTERFACE == 1) &amp;&amp; (OTG_ULPI_CARKIT ==1)</p>
7	GOUTNakEff	R	<p>Mode: Device only Global OUT NAK Effective (GOUTNakEff) 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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not Active</li> <li>■ 0x1 (ACTIVE): Global OUT NAK Effective</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

Bits	Name	Memory Access	Description
6	GINNakEff	R	<p>Mode: Device only  Global IN Non-periodic NAK Effective (GINNakEff)  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 Non-periodic 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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Global Non-periodic IN NAK not active</li> <li>■ 0x1 (ACTIVE): Set Global Non-periodic IN NAK bit</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6 )</p>
5	NPTxFEmp	R	<p>Mode: Host and Device  Non-periodic TxFIFO Empty (NPTxFEmp)  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> <p>In host mode, the application can use GINTSTS.NPTxFEmp with the OTG_ENDED_TX_FIFO parameter set to either 1 or 0.</p> <p>In device mode, the application uses GINTSTS.NPTxFEmp when OTG_ENDED_TX_FIFO=0. When OTG_ENDED_TX_FIFO=1, the application uses DIEPINTn.TxFEmp.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Non-periodic TxFIFO is not empty</li> <li>■ 0x1 (ACTIVE): Non-periodic TxFIFO is empty</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> ((OTG_MODE != 3 &amp;&amp; OTG_MODE != 4)    ((OTG_MODE == 3    OTG_MODE == 4) &amp;&amp; (OTG_ENDED_TX_FIFO == 0)) )</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
4	RxFlvl	R	<p>Mode: Host and Device Rx FIFO Non-Empty (RxFlvl) Indicates that there is at least one packet pending to be read from the Rx FIFO.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Rx Fifo is empty</li> <li>■ 0x1 (ACTIVE): Rx Fifo is not empty</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>
3	Sof	R/W1C	<p>Mode: Host and Device Start of (micro)Frame (Sof) 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, the controller 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. This bit can be set only by the core and the application must write 1 to clear it.</p> <p><b>Note:</b> This register may return 1'b1 if read immediately after power-on reset. If the register bit reads 1'b1 immediately after power-on reset, it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INTACTIVE): No Start of Frame</li> <li>■ 0x1 (ACTIVE): Start of Frame</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

Bits	Name	Memory Access	Description
2	OTGInt	R	<p>Mode: Host and Device OTG Interrupt (OTGInt)</p> <p>The controller 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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): OTG Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>
1	ModeMis	R/W1C	<p>Mode: Host and Device Mode Mismatch Interrupt (ModeMis)</p> <p>The core sets this bit when the application is trying to access:</p> <ul style="list-style-type: none"> <li>■ A Host mode register, when the controller is operating in Device mode</li> <li>■ A Device mode register, when the controller is operating in Host mode</li> </ul> <p>The register access is completed on the AHB with an OKAY response, but is ignored by the controller internally and does not affect the operation of the controller.</p> <p>This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Mode Mismatch Interrupt</li> <li>■ 0x1 (ACTIVE): Mode Mismatch Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>

**Table 7-11 Fields for Register: GINTSTS (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
0	CurMod	R	<p>Mode: Host and Device Current Mode of Operation (CurMod) Indicates the current mode.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Device mode</li> <li>■ 1'b1: Host mode</li> </ul> <p><b>Note:</b> The reset value of this register field can be read only after the PHY clock is stable, or if IDDIG_FILTER is enabled, wait for the filter timer to expire to read the correct reset value which ever event is later.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DEVICE): Device mode</li> <li>■ 0x1 (HOST): Host mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>

### 7.1.7 GINTMSK

- **Name:** Interrupt Mask Register
  - **Description:** This register works with the Interrupt Register (GINTSTS) to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the GINTSTS register bit corresponding to that interrupt is still set.
- Note:** The fields of this register change depending on host or device mode.
- **Size:** 32 bits
  - **Offset:** 0x18
  - **Exists:** Always

WkUpIntMsk	31
SessReqIntMsk	30
DisconnectIntMsk	29
ConnDStsChngMsk	28
LPM_IntMsk	27
PTxFEmpMsk	26
HChIntMsk	25
PrtIntMsk	24
ResetDetiMsk	23
FetSuspMsk	22
incomplPMsK	21
incomplSOINMsk	20
OEPIntMsk	19
IEPIntMsk	18
EPMisMsk	17
RstrDoneIntMsk	16
EOPFMsk	15
ISOOutDropMsk	14
EnumDoneMsk	13
USBRstMsk	12
USBSuspMsk	11
ErySuspMsk	10
I2CIntMsk	9
ULPICKINTMsk	8
GOUTNakEffMsk	7
GINNakEffMsk	6
NPTxFEmpMsk	5
RxFLviMsk	4
SofMsk	3
OTGIntMsk	2
ModeMisMsk	1
RESERVED	0

Table 7-12 Fields for Register: GINTMSK

Bits	Name	Memory Access	Description
31	WkUpIntMsk	R/W	<p>Mode: Host and Device</p> <p>Resume/Remote Wakeup Detected Interrupt Mask (WkUpIntMsk)</p> <p>The WakeUp bit is used for LPM state wake up in a way similar to that of wake up in suspend state.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Resume or Remote Wakeup Detected Interrupt Mask</li> <li>■ 0x1 (NOMASK): Unmask Resume Remote Wakeup Detected Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-12 Fields for Register: GINTMSK (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
30	SessReqIntMsk	R/W	<p>Mode: Host and Device Session Request/New Session Detected Interrupt Mask (SessReqIntMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Session Request or New Session Detected Interrupt Mask</li> <li>■ 0x1 (NOMASK): No Session Request or New Session Detected Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
29	DisconnectIntMsk	R/W	<p>Mode: Host and Device Disconnect Detected Interrupt Mask (DisconnectIntMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Disconnect Detected Interrupt Mask</li> <li>■ 0x1 (NOMASK): No Disconnect Detected Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
28	ConIDStsChngMsk	R/W	<p>Mode: Host and Device Connector ID Status Change Mask (ConIDStsChngMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Connector ID Status Change Mask</li> <li>■ 0x1 (NOMASK): No Connector ID Status Change Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
27	LPM_IntMsk	R/W	<p>Mode: Host and Device LPM Transaction Received Interrupt (LPM_Int) LPM Transaction received interrupt Mask</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): LPM Transaction received interrupt Mask</li> <li>■ 0x1 (NOMASK): No LPM Transaction received interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM==1</p>

**Table 7-12 Fields for Register: GINTMSK (Continued)**

Bits	Name	Memory Access	Description
26	PTxFEmpMsk	R/W	<p>Mode: Host only Periodic TxFIFO Empty Mask (PTxFEmpMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Periodic TxFIFO Empty Mask</li> <li>■ 0x1 (NOMASK): No Periodic TxFIFO Empty Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4) &amp;&amp; OTG_EN_PERIO_HOST ==1</p>
25	HChIntMsk	R/W	<p>Mode: Host only Host Channels Interrupt Mask (HChIntMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Host Channels Interrupt Mask</li> <li>■ 0x1 (NOMASK): No Host Channels Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4)</p>
24	PrtIntMsk	R/W	<p>Mode: Host only Host Port Interrupt Mask (PrtIntMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Host Port Interrupt Mask</li> <li>■ 0x1 (NOMASK): No Host Port Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4)</p>
23	ResetDetMsk	R/W	<p>Mode: Device only Reset detected Interrupt Mask (ResetDetMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Reset detected Interrupt Mask</li> <li>■ 0x1 (NOMASK): No Reset detected Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>
22	FetSuspMsk	R/W	<p>Mode: Device only Data Fetch Suspended Mask (FetSuspMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Data Fetch Suspended Mask</li> <li>■ 0x1 (NOMASK): No Data Fetch Suspended Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>

**Table 7-12 Fields for Register: GINTMSK (Continued)**

Bits	Name	Memory Access	Description
21	incomlPMsK	R/W	<p><b>Incomplete Periodic Transfer Mask (incomlPMsK)</b>  Mode: Host only  <b>Incomplete Isochronous OUT Transfer Interrupt Mask (incomlSOOUTMsk)</b>  Mode: Device only  <b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK):  Host mode: Incomplete Periodic Transfer Mask  Device mode: Incomplete Isochronous OUT Transfer Mask</li> <li>■ 0x1 (NOMASK):  Host mode: No Incomplete Periodic Transfer Mask  Device mode: No Incomplete Isochronous OUT Transfer Mask</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
20	incomlSOINMsk	R/W	<p>Mode: Device only  Incomplete Isochronous IN Transfer Mask (incomlSOINMsk)  This bit is enabled only when device periodic endpoints are enabled in Dedicated TxFIFO mode.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Incomplete Isochronous IN Transfer Mask</li> <li>■ 0x1 (NOMASK): No Incomplete Isochronous IN Transfer Mask</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> (OTG_MODE != 5    OTG_MODE != 6) &amp;&amp; OTG_NUM_PERIO_EPS != 0 &amp;&amp; OTG_EN_DED_TX_FIFO ==1  <b>Testable:</b> writeAsRead</p>
19	OEPIntMsk	R/W	<p>Mode: Device only  OUT Endpoints Interrupt Mask (OEPIntMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): OUT Endpoints Interrupt Mask</li> <li>■ 0x1 (NOMASK): No OUT Endpoints Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>

**Table 7-12 Fields for Register: GINTMSK (Continued)**

Bits	Name	Memory Access	Description
18	IEPIntMsk	R/W	<p>Mode: Device only IN Endpoints Interrupt Mask (IEPIntMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): IN Endpoints Interrupt Mask</li> <li>■ 0x1 (NOMASK): No IN Endpoints Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>
17	EPMisMsk	R/W	<p>Mode: Device only Endpoint Mismatch Interrupt Mask (EPMisMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Endpoint Mismatch Interrupt Mask</li> <li>■ 0x1 (NOMASK): No Endpoint Mismatch Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>
16	RstrDoneIntMsk	R/W	<p>Mode: Host and Device Restore Done Interrupt Mask (RstrDoneIntMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Restore Done Interrupt Mask</li> <li>■ 0x1 (NOMASK): No Restore Done Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_PWROPT == 2</p>
15	EOPFMsks	R/W	<p>Mode: Device only End of Periodic Frame Interrupt Mask (EOPFMsks)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): End of Periodic Frame Interrupt Mask</li> <li>■ 0x1 (NOMASK): No End of Periodic Frame Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>
14	ISOOutDropMsk	R/W	<p>Mode: Device only Isochronous OUT Packet Dropped Interrupt Mask (ISOOutDropMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Isochronous OUT Packet Dropped Interrupt Mask</li> <li>■ 0x1 (NOMASK): No Isochronous OUT Packet Dropped Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>

**Table 7-12 Fields for Register: GINTMSK (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
13	EnumDoneMsk	R/W	<p>Mode: Device only Enumeration Done Mask (EnumDoneMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Enumeration Done Mask</li> <li>■ 0x1 (NOMASK): No Enumeration Done Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>
12	USBRstMsk	R/W	<p>Mode: Device only USB Reset Mask (USBRstMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): USB Reset Mask</li> <li>■ 0x1 (NOMASK): No USB Reset Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>
11	USBSuspMsk	R/W	<p>Mode: Device only USB Suspend Mask (USBSuspMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): USB Suspend Mask</li> <li>■ 0x1 (NOMASK): No USB Suspend Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>
10	ErlySuspMsk	R/W	<p>Mode: Device only Early Suspend Mask (ErlySuspMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Early Suspend Mask</li> <li>■ 0x1 (NOMASK): No Early Suspend Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>
9	I2CIntMsk	R/W	<p>Mode: Host and Device I2C Interrupt Mask (I2CIntMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): I2C Interrupt Mask</li> <li>■ 0x1 (NOMASK): No I2C Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_I2C_INTERFACE == 1)</p>

**Table 7-12 Fields for Register: GINTMSK (Continued)**

Bits	Name	Memory Access	Description
8	ULPICKINTMsk	R/W	<p>Mode: Host and Device ULPI Carkit Interrupt Mask (ULPICKINTMsk) I2C Carkit Interrupt Mask (I2CCKINTMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): ULPI Carkit Interrupt Mask or I2C Carkit Interrupt Mask</li> <li>■ 0x1 (NOMASK): No ULPI Carkit Interrupt Mask or I2C Carkit Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_I2C_INTERFACE == 1) &amp;&amp; (OTG_ULPI_CARKIT ==1)</p>
7	GOUTNakEffMsk	R/W	<p>Mode: Device only Global OUT NAK Effective Mask (GOUTNakEffMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Global OUT NAK Effective Mask</li> <li>■ 0x1 (NOMASK): No Global OUT NAK Effective Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>
6	GINNakEffMsk	R/W	<p>Mode: Device only, Global Non-periodic IN NAK Effective Mask (GINNakEffMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Global Non-periodic IN NAK Effective Mask</li> <li>■ 0x1 (NOMASK): No Global Non-periodic IN NAK Effective Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6)</p>
5	NPTxFEmpMsk	R/W	<p>Mode: Host and Device Non-periodic TxFIFO Empty Mask (NPTxFEmpMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Non-periodic TxFIFO Empty Mask</li> <li>■ 0x1 (NOMASK): No Non-periodic TxFIFO Empty Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> ((OTG_ENDED_TX_FIFO == 0)    (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4))</p>

**Table 7-12 Fields for Register: GINTMSK (Continued)**

Bits	Name	Memory Access	Description
4	RxFlvIMsk	R/W	<p>Mode: Host and Device Receive FIFO Non-Empty Mask (RxFlvIMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Receive FIFO Non-Empty Mask</li> <li>■ 0x1 (NOMASK): No Receive FIFO Non-Empty Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
3	SofMsk	R/W	<p>Mode: Host and Device Start of (micro)Frame Mask (SofMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Start of Frame Mask</li> <li>■ 0x1 (NOMASK): No Start of Frame Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
2	OTGIntMsk	R/W	<p>Mode: Host and Device OTG Interrupt Mask (OTGIntMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): OTG Interrupt Mask</li> <li>■ 0x1 (NOMASK): No OTG Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
1	ModeMisMsk	R/W	<p>Mode: Host and Device Mode Mismatch Interrupt Mask (ModeMisMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mode Mismatch Interrupt Mask</li> <li>■ 0x1 (NOMASK): No Mode Mismatch Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
0	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

### 7.1.8 GRXSTSR

- **Name:** Receive Status Debug Read Register
- **Description:** A read to the Receive Status Debug Read register returns the contents of the top of the Receive FIFO.

The receive status contents must be interpreted differently in Host and Device modes. The core ignores the receive status read when the receive FIFO is empty and returns a value of 32'h0000\_0000.

**Note:**

- Use of these fields vary based on whether the core is functioning as a host or a device.
- Do not read this register's reset value before configuring the core because the read value is 'X' in the simulation.

- **Size:** 32 bits
- **Offset:** 0x1c
- **Exists:** Always

Rsvd	31	RESERVED	30:25	24:21	20:17	16:15	14:4	3:0
------	----	----------	-------	-------	-------	-------	------	-----

**Table 7-13 Fields for Register: GRXSTSR**

Bits	Name	Memory Access	Description
31			<b>Reserved Field:</b> Yes
30:25	RESERVED	R	<b>RESERVED</b> <b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead

**Table 7-13 Fields for Register: GRXSTSR (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
24:21	FN	R	<p>Mode: Device only Frame Number (FN)</p> <p>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.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6</p>

**Table 7-13 Fields for Register: GRXSTSR (Continued)**

Bits	Name	Memory Access	Description
20:17	PktSts	R	<p>Packet Status (PktSts) indicates the status of the received packet.</p> <p>In host mode,</p> <ul style="list-style-type: none"> <li>■ 4'b0010: IN data packet received</li> <li>■ 4'b0011: IN transfer completed (triggers an interrupt)</li> <li>■ 4'b0101: Data toggle error (triggers an interrupt)</li> <li>■ 4'b0111: Channel halted (triggers an interrupt)</li> <li>■ Others: Reserved</li> </ul> <p><b>Reset:</b> 4'b0</p> <p>In device mode,</p> <ul style="list-style-type: none"> <li>■ 4'b0001: Global OUT NAK (triggers an interrupt)</li> <li>■ 4'b0010: OUT data packet received</li> <li>■ 4'b0011: OUT transfer completed (triggers an interrupt)</li> <li>■ 4'b0100: SETUP transaction completed (triggers an interrupt)</li> <li>■ 4'b0110: SETUP data packet received</li> <li>■ Others: Reserved</li> </ul> <p><b>Reset:</b> 4'h0</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OUTNAK): Global OUT NAK in device mode (triggers an interrupt)</li> <li>■ 0x2 (INOUTDPRX): IN data packet received in host mode and OUT data packet received in device mode</li> <li>■ 0x3 (INOUTTRCOM): IN or OUT transfer completed in both host and device mode (triggers an interrupt)</li> <li>■ 0x4 (DSETUPCOM): SETUP transaction completed in device mode (triggers an interrupt)</li> <li>■ 0x5 (DTTOG): Data toggle error (triggers an interrupt) in host mode</li> <li>■ 0x6 (DSETUPRX): SETUP data packet received in device mode</li> <li>■ 0x7 (CHHALT): Channel halted in host mode (triggers an interrupt)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>

**Table 7-13 Fields for Register: GRXSTSR (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
16:15	DPID	R	<p>Data PID (DPID) In host mode, indicates the Data PID of the received packet. In device mode, indicates the Data PID of the received OUT data packet.</p> <ul style="list-style-type: none"> <li>■ 2'b00: DATA0</li> <li>■ 2'b10: DATA1</li> <li>■ 2'b01: DATA2</li> <li>■ 2'b11: MDATA</li> </ul> <p><b>Reset:</b> 2'h0 <b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DATA0): DATA0</li> <li>■ 0x2 (DATA1): DATA1</li> <li>■ 0x1 (DATA2): DATA2</li> <li>■ 0x3 (MDATA): MDATA</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
14:4	BCnt	R	<p>Byte Count (BCnt) In host mode, indicates the byte count of the received IN data packet. In device mode, indicates the byte count of the received data packet.</p> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

**Table 7-13 Fields for Register: GRXSTSR (Continued)**

Bits	Name	Memory Access	Description
3:0	ChNum	R	<p><b>Channel Number (ChNum)</b>  Mode: Host only  Indicates the channel number to which the current received packet belongs.</p> <p><b>Endpoint Number (EPNum)</b>  Mode: Device only  Indicates the endpoint number to which the current received packet belongs.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (CHEP0): Channel or EndPoint 0</li> <li>■ 0x1 (CHEP1): Channel or EndPoint 1</li> <li>■ 0x2 (CHEP2): Channel or EndPoint 2</li> <li>■ 0x3 (CHEP3): Channel or EndPoint 3</li> <li>■ 0x4 (CHEP4): Channel or EndPoint 4</li> <li>■ 0x5 (CHEP5): Channel or EndPoint 5</li> <li>■ 0x6 (CHEP6): Channel or EndPoint 6</li> <li>■ 0x7 (CHEP7): Channel or EndPoint 7</li> <li>■ 0x8 (CHEP8): Channel or EndPoint 8</li> <li>■ 0x9 (CHEP9): Channel or EndPoint 9</li> <li>■ 0xa (CHEP10): Channel or EndPoint 10</li> <li>■ 0xb (CHEP11): Channel or EndPoint 11</li> <li>■ 0xc (CHEP12): Channel or EndPoint 12</li> <li>■ 0xd (CHEP13): Channel or EndPoint 13</li> <li>■ 0xe (CHEP14): Channel or EndPoint 14</li> <li>■ 0xf (CHEP15): Channel or EndPoint 15</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>

### 7.1.9 GRXSTSP

- **Name:** Receive Status Read/Pop Register
- **Description:** A read to the Receive Status Read and Pop register returns the contents of the top of the Receive FIFO and additionally pops the top data entry out of the RxFIFO.

The receive status contents must be interpreted differently in Host and Device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 32'h0000\_0000. The application must only pop the Receive Status FIFO when the Receive FIFO Non-Empty bit of the Core Interrupt register (GINTSTS.RxFLvl) is asserted.

**Note:**

- Use of these fields vary based on whether the core is functioning as a host or a device.
- Do not read this register's reset value before configuring the core because the read value is 'X' in the simulation.
- **Size:** 32 bits
- **Offset:** 0x20
- **Exists:** Always

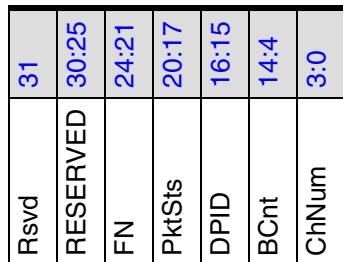


Table 7-14 Fields for Register: GRXSTSP

Bits	Name	Memory Access	Description
31			<b>Reserved Field:</b> Yes
30:25	RESERVED	R	<b>RESERVED</b> <b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead

**Table 7-14 Fields for Register: GRXSTSP (Continued)**

Bits	Name	Memory Access	Description
24:21	FN	R	<p>Mode: Device only Frame Number (FN) 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.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE != 5 &amp;&amp; OTG_MODE != 6</p>
20:17	PktSts	R	<p>Packet Status (PktSts) indicates the status of the received packet.</p> <p>In host mode,</p> <ul style="list-style-type: none"> <li>■ 4'b0010: IN data packet received</li> <li>■ 4'b0011: IN transfer completed (triggers an interrupt)</li> <li>■ 4'b0101: Data toggle error (triggers an interrupt)</li> <li>■ 4'b0111: Channel halted (triggers an interrupt)</li> <li>■ Others: Reserved</li> </ul> <p><b>Reset:</b> 4'b0</p> <p>In device mode,</p> <ul style="list-style-type: none"> <li>■ 4'b0001: Global OUT NAK (triggers an interrupt)</li> <li>■ 4'b0010: OUT data packet received</li> <li>■ 4'b0011: OUT transfer completed (triggers an interrupt)</li> <li>■ 4'b0100: SETUP transaction completed (triggers an interrupt)</li> <li>■ 4'b0110: SETUP data packet received</li> <li>■ Others: Reserved</li> </ul> <p><b>Reset:</b> 4'h0</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OUTNAK): Global OUT NAK in device mode (triggers an interrupt)</li> <li>■ 0x2 (INOUTDPRX): IN data packet received in host mode and OUT data packet received in device mode</li> <li>■ 0x3 (INOUTTRCOM): IN or OUT transfer completed in both host and device mode (triggers an interrupt)</li> <li>■ 0x4 (DSETUPCOM): SETUP transaction completed in device mode (triggers an interrupt)</li> <li>■ 0x5 (DTTOG): Data toggle error (triggers an interrupt) in host mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>

**Table 7-14 Fields for Register: GRXSTSP (Continued)**

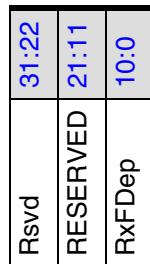
<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
16:15	DPID	R	<p>Data PID (DPID) In host mode, indicates the Data PID of the received packet. In device mode, indicates the Data PID of the received OUT data packet.</p> <ul style="list-style-type: none"> <li>■ 2'b00: DATA0</li> <li>■ 2'b10: DATA1</li> <li>■ 2'b01: DATA2</li> <li>■ 2'b11: MDATA</li> </ul> <p><b>Reset:</b> 2'h0 <b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DATA0): DATA0</li> <li>■ 0x2 (DATA1): DATA1</li> <li>■ 0x1 (DATA2): DATA2</li> <li>■ 0x3 (MDATA): MDATA</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
14:4	BCnt	R	<p>Byte Count (BCnt) In host mode, indicates the byte count of the received IN data packet. In device mode, indicates the byte count of the received data packet.</p> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

**Table 7-14 Fields for Register: GRXSTSP (Continued)**

Bits	Name	Memory Access	Description
3:0	ChNum	R	<p><b>Channel Number (ChNum)</b> Mode: Host only Indicates the channel number to which the current received packet belongs.</p> <p><b>Endpoint Number (EPNum)</b> Mode: Device only Indicates the endpoint number to which the current received packet belongs.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (CHEP0): Channel or EndPoint 0</li> <li>■ 0x1 (CHEP1): Channel or EndPoint 1</li> <li>■ 0x2 (CHEP2): Channel or EndPoint 2</li> <li>■ 0x3 (CHEP3): Channel or EndPoint 3</li> <li>■ 0x4 (CHEP4): Channel or EndPoint 4</li> <li>■ 0x5 (CHEP5): Channel or EndPoint 5</li> <li>■ 0x6 (CHEP6): Channel or EndPoint 6</li> <li>■ 0x7 (CHEP7): Channel or EndPoint 7</li> <li>■ 0x8 (CHEP8): Channel or EndPoint 8</li> <li>■ 0x9 (CHEP9): Channel or EndPoint 9</li> <li>■ 0xa (CHEP10): Channel or EndPoint 10</li> <li>■ 0xb (CHEP11): Channel or EndPoint 11</li> <li>■ 0xc (CHEP12): Channel or EndPoint 12</li> <li>■ 0xd (CHEP13): Channel or EndPoint 13</li> <li>■ 0xe (CHEP14): Channel or EndPoint 14</li> <li>■ 0xf (CHEP15): Channel or EndPoint 15</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>

### 7.1.10 GRXFSIZ

- **Name:** Receive FIFO Size Register
- **Description:** The application can program the RAM size that must be allocated to the RxFIFO.
- **Size:** 32 bits
- **Offset:** 0x24
- **Exists:** Always



**Table 7-15 Fields for Register: GRXFSIZ**

Bits	Name	Memory Access	Description
31:22			<b>Reserved Field:</b> Yes
21:11	RESERVED	R	<b>RESERVED</b> <b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead
10:0	RxFDep	OTG_DFI FO_DYN AMIC == 1 ? R/W : R	Mode: Host and Device Rx FIFO Depth (RxFDep) This value is in terms of 32-bit words. <ul style="list-style-type: none"> <li>■ Minimum value is 16</li> <li>■ Maximum value is 32,768</li> </ul> The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth during configuration. If Enable Dynamic FIFO Sizing is selected in coreConsultant, these flops are optimized, and reads return the power-on value. If Enable Dynamic FIFO Sizing is selected in coreConsultant, you can write a new value in this field. Programmed values must not exceed the power-on value. <b>Value After Reset:</b> 0x400 <b>Exists:</b> Always

### 7.1.11 GNPTXFSIZ

- **Name:** Non-periodic Transmit FIFO Size Register
- **Description:** The application can program the RAM size and the memory start address for the Non-periodic TxFIFO

**Note:** The fields of this register change depending on host or device mode.

- **Size:** 32 bits
- **Offset:** 0x28
- **Exists:** Always



**Table 7-16 Fields for Register: GNPTXFSIZ**

Bits	Name	Memory Access	Description
31:27			<b>Reserved Field:</b> Yes

**Table 7-16 Fields for Register: GNPTXFSIZ (Continued)**

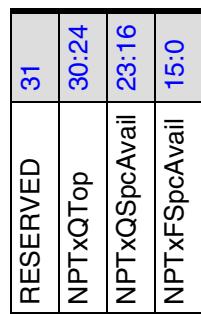
Bits	Name	Memory Access	Description
26:16	NPTxFDep	OTG_DFI FO_DYN AMIC == 1 ? R/W : R	<p><b>Non-periodic TxFIFO Depth (NPTxFDep)</b></p> <p>The NPTxFDep field description is valid only for,</p> <ul style="list-style-type: none"> <li>■ host mode</li> <li>■ device mode when OTG_ENDED_TX_FIFO=0</li> </ul> <p>This value is in terms of 32-bit words.</p> <ul style="list-style-type: none"> <li>■ Minimum value is 16</li> <li>■ Maximum value is 32,768</li> </ul> <p>This attribute of field is determined during coreConsultant configuration by "Enable Dynamic FIFO Sizing?" (OTG_DFIFO_DYNAMIC):</p> <ul style="list-style-type: none"> <li>■ OTG_DFIFO_DYNAMIC = 0: These flops are optimized, and reads return the power-on value.</li> <li>■ OTG_DFIFO_DYNAMIC = 1: The application can write a new value in this field. Programmed values must not exceed the power-on value set in coreConsultant.</li> </ul> <p>The power-on reset value of this field is specified during coreConsultant configuration:</p> <ul style="list-style-type: none"> <li>■ Host: <ul style="list-style-type: none"> <li>- The reset value is OTG_TX_HNPERIO_DFIFO_DEPTH parameter when OTG_ENDED_TX_FIFO == 0.</li> <li>- The reset value is OTG_TX_DINEP_DFIFO_DEPTH_0 parameter when OTG_ENDED_TX_FIFO == 1.</li> </ul> </li> <li>■ Device Shared FIFO mode: The reset value is the Largest Non-periodic Tx Data FIFO Depth parameter, OTG_TX_NPERIO_DFIFO_DEPTH, when OTG_ENDED_TX_FIFO == 0.</li> </ul> <p><b>IN Endpoint TxFIFO 0 Depth (INEPTxF0Dep)</b></p> <p>The INEPTxF0Dep field description is valid only for device mode when OTG_ENDED_TX_FIFO=1. This value is in terms of 32-bit words.</p> <ul style="list-style-type: none"> <li>■ Minimum value is 16</li> <li>■ Maximum value is 32,768</li> </ul> <p>The attribute of this field is determined during coreConsultant configuration by Enable Dynamic FIFO Sizing? (OTG_DFIFO_DYNAMIC):</p> <ul style="list-style-type: none"> <li>■ OTG_DFIFO_DYNAMIC = 0: These flops are optimized, and reads return the power-on value.</li> </ul>

**Table 7-16 Fields for Register: GNPTXFSIZ (Continued)**

Bits	Name	Memory Access	Description
			<ul style="list-style-type: none"> <li>■ OTG_DFIFO_DYNAMIC = 1: The application can write a new value in this field. Programmed values must not exceed the power-on value set in coreConsultant.</li> </ul> <p>The power-on reset value of this field is specified during coreConsultant configuration as Largest IN Endpoint FIFO 0 Depth (parameter OTG_TX_DINEP_DFIFO_DEPTH_0).</p> <p><b>Value After Reset:</b> 0x400</p> <p><b>Exists:</b> Always</p>
15:11			<b>Reserved Field:</b> Yes
10:0	NPTxFStAddr	OTG_DFIFO_DYN_AMIC == 1 ? R/W : R	<p><b>Non-periodic Transmit RAM Start Address (NPTxFStAddr)</b></p> <p>The NPTxFStAddr field description is valid only for host mode. This field contains the memory start address for Non-periodic Transmit FIFO RAM.</p> <p>This field is determined during coreConsultant configuration by Enable Dynamic FIFO Sizing?(OTG_DFIFO_DYNAMIC).</p> <ul style="list-style-type: none"> <li>■ OTG_DFIFO_DYNAMIC = 0: These flops are optimized, and reads return the power-on value.</li> <li>■ OTG_DFIFO_DYNAMIC = 1: The application can write a new value in this field. Programmed values must not exceed the power-on value set in coreConsultant.</li> </ul> <p>Programmed values must not exceed the power-on value set in coreConsultant.</p> <p>The power-on reset value of this field is specified during coreConsultant configuration by Largest Rx Data FIFO Depth (parameter OTG_RX_DFIFO_DEPTH).</p> <p><b>IN Endpoint FIFO0 Transmit RAM Start Address(INEPTxF0StAddr)</b></p> <p>The INEPTxF0StAddr field description is valid only for device mode when OTG_ENDED_TX_FIFO=1.</p> <p>This field contains the memory start address for IN Endpoint Transmit FIFO# 0.</p> <p>Programmed values must not exceed the power-on value.</p> <p><b>Value After Reset:</b> 0x400</p> <p><b>Exists:</b> Always</p>

### 7.1.12 GNPTXSTS

- **Name:** Non-periodic Transmit FIFO/Queue Status Register
- **Description:** In Device mode, this register is valid only in Shared FIFO operation.  
This read-only register contains the free space information for the Non-periodic TxFIFO and the Non-periodic Transmit Request Queue.
- **Size:** 32 bits
- **Offset:** 0x2c
- **Exists:** (OTG\_MODE !=3 && OTG\_MODE !=4) || (OTG\_ENDED\_TX\_FIFO == 0)



**Table 7-17 Fields for Register: GNPTXSTS**

Bits	Name	Memory Access	Description
31	RESERVED	R	<b>RESERVED</b> <b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead

**Table 7-17 Fields for Register: GNPTXSTS (Continued)**

Bits	Name	Memory Access	Description
30:24	NPTxQTop	R	<p>Top of the Non-periodic Transmit Request Queue (NPTxQTop) Entry in the Non-periodic Tx Request Queue that is currently being processed by the MAC.</p> <ul style="list-style-type: none"> <li>■ Bits [30:27]: Channel/endpoint number</li> <li>■ Bits [26:25]:</li> <li>■ 2'b00: IN/OUT token <ul style="list-style-type: none"> <li>- 2'b01: Zero-length transmit packet (device IN/host OUT)</li> <li>- 2'b10: PING/CSPLIT token</li> <li>- 2'b11: Channel halt command</li> </ul> </li> <li>■ Bit [24]: Terminate (last Entry for selected channel/endpoint)</li> </ul> <p><b>Reset:</b> 7'h0</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INOUTTK): IN/OUT token</li> <li>■ 0x1 (ZEROTX): Zero-length transmit packet (device IN/host OUT)</li> <li>■ 0x2 (PINGCSPLIT): PING/CSPLIT token</li> <li>■ 0x3 (CHNHALT): Channel halt command</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> writeAsRead</p>

**Table 7-17 Fields for Register: GNPTXSTS (Continued)**

Bits	Name	Memory Access	Description
23:16	NPTxQSpAvail	R	<p>Non-periodic Transmit Request Queue Space Available (NPTxQSpAvail)      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> <ul style="list-style-type: none"> <li>■ 8'h0: Non-periodic Transmit Request Queue is full</li> <li>■ 8'h1: 1 location available</li> <li>■ 8'h2: 2 locations available</li> <li>■ n: n locations available (0 &lt;= n &lt;= 8)</li> <li>■ Others: Reserved</li> </ul> <p><b>Reset:</b> Configurable  <b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (FULL): Non-periodic Transmit Request Queue is full</li> <li>■ 0x1 (QUE1): 1 location available</li> <li>■ 0x2 (QUE2): 2 locations available</li> <li>■ 0x3 (QUE3): 3 locations available</li> <li>■ 0x4 (QUE4): 4 locations available</li> <li>■ 0x5 (QUE5): 5 locations available</li> <li>■ 0x6 (QUE6): 6 locations available</li> <li>■ 0x7 (QUE7): 7 locations available</li> <li>■ 0x8 (QUE8): 8 locations available</li> </ul> <p><b>Value After Reset:</b> 0x8  <b>Exists:</b> Always  <b>Testable:</b> writeAsRead</p>
15:0	NPTxFSpAvail	R	<p>Non-periodic TxFIFO Space Avail (NPTxFSpAvail)      Indicates the amount of free space available in the Non-periodic TxFIFO.      Values are in terms of 32-bit words.</p> <ul style="list-style-type: none"> <li>■ 16'h0: Non-periodic TxFIFO is full</li> <li>■ 16'h1: 1 word available</li> <li>■ 16'h2: 2 words available</li> <li>■ 16'hn: n words available (where 0 &lt;= n &lt;= 32,768)</li> <li>■ 16'h8000: 32,768 words available</li> <li>■ Others: Reserved</li> </ul> <p><b>Reset:</b> Configurable  <b>Value After Reset:</b> 0x400  <b>Exists:</b> Always  <b>Testable:</b> writeAsRead</p>

### 7.1.13 GI2CCTL

- **Name:** I2C Access Register
- **Description:** The application can use this register to access OTG devices connected to the OTG core through the I2C interface. It is implemented only if Enable I2C Interface was selected during coreConsultant configuration (parameter OTG\_I2C\_INTERFACE = 1). The I2C interface on the OTG core can read/write the register space in the attached I2C device. The following table describes the register fields.
- **Size:** 32 bits
- **Offset:** 0x30
- **Exists:** OTG\_I2C\_INTERFACE == 1

BsyDne	31	RW	30	RESERVED	29	I2CDatSe0	28	I2CDevAdr	27:26	I2CSuspCtl	25	Ack	24	I2CEn	23	Addr	22:16	RegAddr	15:8	RWData	7:0
--------	----	----	----	----------	----	-----------	----	-----------	-------	------------	----	-----	----	-------	----	------	-------	---------	------	--------	-----

Table 7-18 Fields for Register: GI2CCTL

Bits	Name	Memory Access	Description
31	BsyDne	R/W1S	<p>I2C Busy/Done (BsyDne)            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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DONE): I2C interface is done with the transfer</li> <li>■ 0x1 (BUSY): I2C interface is busy with the transfer</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_I2C_INTERFACE == 1</p>

**Table 7-18 Fields for Register: GI2CCTL (Continued)**

Bits	Name	Memory Access	Description
30	RW	R/W	<p>Read/Write Indicator (RW) Indicates whether a read or write register transfer must be performed on the interface. Read/write bursting is not supported for registers. 1'b0: Write 1'b1: Read</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (WRITE): Performing write transfer</li> <li>■ 0x1 (READ): Performing read transfer</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_I2C_INTERFACE == 1</p>
29	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
28	I2CDatSe0	R/W	<p>I2C DatSe0 USB Mode (I2CDatSe0) Selects the FS interface USB mode.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTACTIVE): Selected VP_VM USB mode</li> <li>■ 0x1 (ACTIVE): Selected DAT_SE0 USB mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_I2C_INTERFACE == 1</p>
27:26	I2CDevAdr	R/W	<p>I2C Device Address (I2CDevAdr) Selects the address of the I2C Slave on the USB 1.1 full-speed serial transceiver that the core uses for OTG signaling.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ADDR1): Selected address of I2C Slave is 2C</li> <li>■ 0x1 (ADDR2): Selected address of I2C Slave is 2D</li> <li>■ 0x2 (ADDR3): Selected address of I2C Slave is 2E</li> <li>■ 0x3 (ADDR4): Selected address of I2C Slave is 2F</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_I2C_INTERFACE == 1</p>

**Table 7-18 Fields for Register: GI2CCTL (Continued)**

Bits	Name	Memory Access	Description
25	I2CSuspCtl	R/W	<p>I2C Suspend Control (I2CSuspCtl) Selects how Suspend is connected to a full-speed transceiver in I2C mode. When I2CSuspCtl bit is set(1'b1), even during suspend the controller should be provide with a running clock to indicate the suspend exit. The controller generates I2C clock from the utmifs_clk. If the clock to the controller is gated during suspend, the controller cannot write the suspend exit to I2C registers.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTACTIVE): Dedicated utmi_suspend_n pin is used</li> <li>■ 0x1 (ACTIVE): Use an I2C write to program the Suspend bit in the PHY register</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_I2C_INTERFACE == 1</p>
24	Ack	R	<p>I2C ACK (Ack)</p> <p>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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NAK): NAK response is received from I2C Slave</li> <li>■ 0x1 (ACK): ACK response is received from I2C Slave</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> OTG_I2C_INTERFACE == 1</p>
23	I2CEn	R/W	<p>I2C Enable (I2CEn)</p> <p>Enables the I2C Master to initiate I2C transactions on the I2C interface.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Not active</li> <li>■ 0x1 (ENABLE): Enables the I2C Master to initiate I2C transactions on the I2C interface.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_I2C_INTERFACE == 1</p>
22:16	Addr	R/W	<p>I2C Address (Addr) 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.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_I2C_INTERFACE == 1</p>

**Table 7-18 Fields for Register: GI2CCTL (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
15:8	RegAddr	R/W	I2C Register Addr (RegAddr) This field programs the address of the register to be read from or written to. <b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_I2C_INTERFACE == 1
7:0	RWData	R/W	I2C Read/Write Data (RWData) 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. <b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_I2C_INTERFACE == 1

### 7.1.14 GPVNDCTL

- **Name:** PHY Vendor Control Register
- **Description:** The application can use this register to access PHY registers. It is implemented only if Enable PHY Vendor Control Interface was selected during coreConsultant configuration (parameter OTG\_VENDOR\_CTL\_INTERFACE = 1). For a UTMI+ PHY, the DWC\_otg core uses the UTMI+ Vendor Control interface for PHY register access. For a ULPI PHY, the core uses the ULPI interface for PHY register access. The application sets Vendor Control register for PHY register access and times the PHY register access. The application polls the VStatus Done bit in this register for the completion of the PHY register access.
- **Size:** 32 bits
- **Offset:** 0x34
- **Exists:** OTG\_VENDOR\_CTL\_INTERFACE == 1

31	DisUlpiDrv
30:28	RESERVED
27	VStsDone
26	VStsBsy
25	NewRegReq
24:23	RESERVED1
22	RegWr
21:16	RegAddr
15:8	VCtrl
7:0	RegData

Table 7-19 Fields for Register: GPVNDCTL

Bits	Name	Memory Access	Description
31	DisUlpiDrv	R/W1S	<p>Disable ULPI Drivers (DisUlpiDrv)</p> <p>The application sets this bit when it has finished processing the ULPI Carkit Interrupt (GINTSTS.ULPICKINT). When set, the controller disables drivers for output signals and masks input signal for the ULPI interface. The controller clears this bit before enabling the ULPI interface.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ENABLED): Enable ULPI ouput signals</li> <li>■ 0x1 (DISABLED): Disable ULPI ouput signals</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_VENDOR_CTL_INTERFACE == 1 &amp;&amp; OTG_ULPI_CARKIT == 1</p>

**Table 7-19 Fields for Register: GPVNDCTL (Continued)**

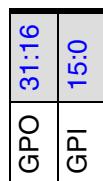
Bits	Name	Memory Access	Description
30:28	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
27	VStsDone	R/W1C	<p>VStatus Done (VStsDone)</p> <p>The core sets this bit when the vendor control access is done. This bit is cleared by the core when the application sets the New Register Request bit (bit 25).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): VStatus Done inactive</li> <li>■ 0x1 (ACTIVE): VStatus Done active</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_VENDOR_CTL_INTERFACE == 1</p>
26	VStsBsy	R	<p>VStatus Busy (VStsBsy)</p> <p>The core sets this bit when the vendor control access is in progress and clears this bit when done.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): VStatus Busy inactive</li> <li>■ 0x1 (ACTIVE): VStatus Busy active</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_VENDOR_CTL_INTERFACE == 1</p> <p><b>Volatile:</b> true</p>
25	NewRegReq	R/W1S	<p>New Register Request (NewRegReq)</p> <p>The application sets this bit for a new vendor control access.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): New Register Request not active</li> <li>■ 0x1 (ACTIVE): New Register Request active</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_VENDOR_CTL_INTERFACE == 1</p> <p><b>Volatile:</b> true</p>
24:23	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-19 Fields for Register: GPVNDCTL (Continued)**

Bits	Name	Memory Access	Description
22	RegWr	R/W	<p>Register Write (RegWr) Set this bit for register writes, and clear it for register reads.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (READ): Register Read</li> <li>■ 0x1 (WRITE): Register Write</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_VENDOR_CTL_INTERFACE == 1</p>
21:16	RegAddr	R/W	<p>Register Address (RegAddr)</p> <p>The 6-bit PHY register address for immediate PHY Register set access. Set to 6'h2F for Extended PHY Register set access.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_VENDOR_CTL_INTERFACE == 1</p>
15:8	VCtrl	R/W	<p>UTMI+ Vendor Control Register Address (VCtrl) 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.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_VENDOR_CTL_INTERFACE == 1</p>
7:0	RegData	R/W	<p>Register Data (RegData)</p> <p>Contains the write data for register write.</p> <p>Read data for register read, valid when VStatus Done is set.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_VENDOR_CTL_INTERFACE == 1</p>

### 7.1.15 GPIO

- **Name:** General Purpose Input/Output Register
- **Description:** The application can use this register for general purpose input/output ports or for debugging.
- **Size:** 32 bits
- **Offset:** 0x38
- **Exists:** OTG\_RM\_OPT\_FEATURES == 0



**Table 7-20 Fields for Register: GPIO**

Bits	Name	Memory Access	Description
31:16	GPO	R/W	<p>General Purpose Output (GPO)</p> <p>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.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_RM_OPT_FEATURES == 0</p>
15:0	GPI	R	<p>General Purpose Input (GPI)</p> <p>This field's read value reflects the gp_i[15:0] core input value.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_RM_OPT_FEATURES == 0</p>

### 7.1.16 GUID

- **Name:** User ID Register
- **Description:** This is a read/write register containing the User ID. It is implemented only if Remove Optional Features? was deselected during coreConsultant configuration (parameter OTG\_RM\_OPT\_FEATURES = 0). The power-on value for this register is specified as the Power-on Value of User ID Register User Identification Register during coreConsultant configuration (parameter OTG\_USERID). This register can be used in the following ways:
  - To store the version or revision of your system
  - To store hardware configurations that are outside the DWC\_otg core
  - As a scratch register
- **Size:** 32 bits
- **Offset:** 0x3c
- **Exists:** OTG\_RM\_OPT\_FEATURES == 0



**Table 7-21 Fields for Register: GUID**

Bits	Name	Memory Access	Description
31:0	GUID	R/W	User ID (UserID) Application-programmable ID field. Reset value is configurable. <b>Value After Reset:</b> 0x12345678 <b>Exists:</b> Always

### 7.1.17 GSNPSID

- **Name:** Synopsys ID Register
- **Description:** This read-only register contains the release number of the core being used.
- **Size:** 32 bits
- **Offset:** 0x40
- **Exists:** Always

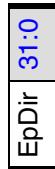


**Table 7-22 Fields for Register: GSNPSID**

Bits	Name	Memory Access	Description
31:0	SynopsysID	R	<p>Release number of the controller being used currently.</p> <p>Bits [31:16]:</p> <ul style="list-style-type: none"> <li>■ 0x4f54: ASCII Value for OT</li> </ul> <p>Bits [15:0]: Current Release Number. For example, 0x400a corresponds to v4.00a Release number.</p> <p><b>Value After Reset:</b> 0x4f54420a</p> <p><b>Exists:</b> Always</p>

### 7.1.18 GHWCFG1

- **Name:** User Hardware Configuration 1 Register
- **Description:** This register contains the logical endpoint direction(s) selected using coreConsultant.
- **Size:** 32 bits
- **Offset:** 0x44
- **Exists:** Always



**Table 7-23 Fields for Register: GHWCFG1**

Bits	Name	Memory Access	Description
31:0	EpDir	R	<p>This 32-bit field uses two bits per endpoint to determine the endpoint direction.</p> <p>Endpoint</p> <ul style="list-style-type: none"> <li>■ Bits [31:30]: Endpoint 15 direction</li> <li>■ Bits [29:28]: Endpoint 14 direction</li> <li>...</li> <li>■ Bits [3:2]: Endpoint 1 direction</li> <li>■ Bits[1:0]: Endpoint 0 direction (always BIDIR)</li> </ul> <p>Direction</p> <ul style="list-style-type: none"> <li>■ 2'b00: BIDIR (IN and OUT) endpoint</li> <li>■ 2'b01: IN endpoint</li> <li>■ 2'b10: OUT endpoint</li> <li>■ 2'b11: Reserved</li> </ul> <p><b>Note:</b> This field is configured using the OTG_EP_DIR_1(n) parameter.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

7.1.19 GHWCFG2

- **Name:** User Hardware Configuration 2 Register
  - **Description:** This register contains configuration options selected using coreConsultant.
  - **Size:** 32 bits
  - **Offset:** 0x48
  - **Exists:** Always

OTG_ENABLE_IC_USB	31
TknQDepth	30:26
PTxQDepth	25:24
NPTxQDepth	23:22
RESERVED	21
MulticProclntrpt	20
DynFifoSizing	19
PerioSupport	18
NumHstChnl	17:14
NumDevEps	13:10
FSPhyType	9:8
HSPhyType	7:6
SingPnt	5
OtgArch	4:3
OtgMode	2:0

**Table 7-24 Fields for Register: GHWCFG2**

Bits	Name	Memory Access	Description
31	OTG_ENABLE_IC_USB	R	<p>IC_USB mode specified for mode of operation</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Disabled the IC_USB Full-Speed Serial Transceiver interface</li> <li>■ 0x1 (ENABLE): Enabled the IC_USB Full-Speed Serial Transceiver interface</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> OTG_FSPHY_INTERFACE == 1 &amp;&amp; OTG_ENABLE_IC_USB == 1</p>
30:26	TknQDepth	R	<p>Device Mode IN Token Sequence Learning Queue Depth (TknQDepth)</p> <p>Range: 0-30</p> <p><b>Note:</b> This field is configured using the OTG_TOKEN_QUEUE_DEPTH parameter.</p> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> Always</p>

**Table 7-24 Fields for Register: GHWCFG2 (Continued)**

Bits	Name	Memory Access	Description
25:24	PTxQDepth	R	<p>Host Mode Periodic Request Queue Depth (PTxQDepth)</p> <ul style="list-style-type: none"> <li>■ 2'b00: 2</li> <li>■ 2'b01: 4</li> <li>■ 2'b10: 8</li> <li>■ 2'b11:16</li> </ul> <p><b>Note:</b> This field is configured using the OTG_PERIO_TX_QUEUE_DEPTH parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (QUE2): Queue Depth 2</li> <li>■ 0x1 (QUE4): Queue Depth 4</li> <li>■ 0x2 (QUE8): Queue Depth 8</li> <li>■ 0x3 (QUE16): Queue Depth 16</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> Always</p>
23:22	NPTxQDepth	R	<p>Non-periodic Request Queue Depth (NPTxQDepth)</p> <ul style="list-style-type: none"> <li>■ 2'b00: 2</li> <li>■ 2'b01: 4</li> <li>■ 2'b10: 8</li> <li>■ Others: Reserved</li> </ul> <p><b>Note:</b> This field is configured using the OTG_NPERIO_TX_QUEUE_DEPTH parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (TWO): Queue size 2</li> <li>■ 0x1 (FOUR): Queue size 4</li> <li>■ 0x2 (EIGHT): Queue size 8</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> Always</p>
21	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-24 Fields for Register: GHWCFG2 (Continued)**

Bits	Name	Memory Access	Description
20	MultiProcIntrpt	R	<p>Multi Processor Interrupt Enabled (MultiProcIntrpt)</p> <ul style="list-style-type: none"> <li>■ 1'b0: No</li> <li>■ 1'b1: Yes</li> </ul> <p><b>Note:</b> This field is configured using the OTG_MULTI_PROC_INTRPT parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No Multi Processor Interrupt Enabled</li> <li>■ 0x1 (ENABLED): Multi Processor Interrupt Enabled</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> Always</p>
19	DynFifoSizing	R	<p>Dynamic FIFO Sizing Enabled (DynFifoSizing)</p> <ul style="list-style-type: none"> <li>■ 1'b0: No</li> <li>■ 1'b1: Yes</li> </ul> <p><b>Note:</b> This field is configured using the OTG_DFIFO_DYNAMIC parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Dynamic FIFO Sizing Disabled</li> <li>■ 0x1 (ENABLED): Dynamic FIFO Sizing Enabled</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> Always</p>
18	PerioSupport	R	<p>Periodic OUT Channels Supported in Host Mode (PerioSupport)</p> <ul style="list-style-type: none"> <li>■ 1'b0: No</li> <li>■ 1'b1: Yes</li> </ul> <p><b>Note:</b> This field is configured using the OTG_EN_PERIO_HOST parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Periodic OUT Channels is not supported in Host Mode</li> <li>■ 0x1 (ENABLED): Periodic OUT Channels Supported in Host Mode Supported</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> Always</p>

**Table 7-24 Fields for Register: GHWCFG2 (Continued)**

Bits	Name	Memory Access	Description
17:14	NumHstChnl	R	<p>Number of Host Channels (NumHstChnl)      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.</p> <p><b>Note:</b> This field is configured using the OTG_NUM_HOST_CHAN parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (HOSTCH0): Host Channel 1</li> <li>■ 0x1 (HOSTCH1): Host Channel 2</li> <li>■ 0x2 (HOSTCH2): Host Channel 3</li> <li>■ 0x3 (HOSTCH3): Host Channel 4</li> <li>■ 0x4 (HOSTCH4): Host Channel 5</li> <li>■ 0x5 (HOSTCH5): Host Channel 6</li> <li>■ 0x6 (HOSTCH6): Host Channel 7</li> <li>■ 0x7 (HOSTCH7): Host Channel 8</li> <li>■ 0x8 (HOSTCH8): Host Channel 9</li> <li>■ 0x9 (HOSTCH9): Host Channel 10</li> <li>■ 0xa (HOSTCH10): Host Channel 11</li> <li>■ 0xb (HOSTCH11): Host Channel 12</li> <li>■ 0xc (HOSTCH12): Host Channel 13</li> <li>■ 0xd (HOSTCH13): Host Channel 14</li> <li>■ 0xe (HOSTCH14): Host Channel 15</li> <li>■ 0xf (HOSTCH15): Host Channel 16</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration  <b>Exists:</b> Always</p>

**Table 7-24 Fields for Register: GHWCFG2 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
13:10	NumDevEps	R	<p>Number of Device Endpoints (NumDevEps)      Indicates the number of device endpoints supported by the core in Device mode.      The range of this field is 0-15.</p> <p><b>Note:</b> This field is configured using the OTG_NUM_EPS parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ENDPT0): End point 0</li> <li>■ 0x1 (ENDPT1): End point 1</li> <li>■ 0x2 (ENDPT2): End point 2</li> <li>■ 0x3 (ENDPT3): End point 3</li> <li>■ 0x4 (ENDPT4): End point 4</li> <li>■ 0x5 (ENDPT5): End point 5</li> <li>■ 0x6 (ENDPT6): End point 6</li> <li>■ 0x7 (ENDPT7): End point 7</li> <li>■ 0x8 (ENDPT8): End point 8</li> <li>■ 0x9 (ENDPT9): End point 9</li> <li>■ 0xa (ENDPT10): End point 10</li> <li>■ 0xb (ENDPT11): End point 11</li> <li>■ 0xc (ENDPT12): End point 12</li> <li>■ 0xd (ENDPT13): End point 13</li> <li>■ 0xe (ENDPT14): End point 14</li> <li>■ 0xf (ENDPT15): End point 15</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration  <b>Exists:</b> Always</p>

**Table 7-24 Fields for Register: GHWCFG2 (Continued)**

Bits	Name	Memory Access	Description
9:8	FSPhyType	R	<p>Full-Speed PHY Interface Type (FSPhyType)</p> <ul style="list-style-type: none"> <li>■ 2'b00: Full-speed interface not supported</li> <li>■ 2'b01: Dedicated full-speed interface</li> <li>■ 2'b10: FS pins shared with UTMI+ pins</li> <li>■ 2'b11: FS pins shared with ULPI pins</li> </ul> <p><b>Note:</b> This field is configured using the OTG_FSPHY_INTERFACE parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NO_FS): Full-speed interface not supported</li> <li>■ 0x1 (FS): Dedicated full-speed interface is supported</li> <li>■ 0x2 (FSPLUSUTMI): FS pins shared with UTMI+ pins is supported</li> <li>■ 0x3 (FSPLUSULPI): FS pins shared with ULPI pins is supported</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> Always</p>
7:6	HSPhyType	R	<p>High-Speed PHY Interface Type (HSPhyType)</p> <ul style="list-style-type: none"> <li>■ 2'b00: High-Speed interface not supported</li> <li>■ 2'b01: UTMI+</li> <li>■ 2'b10: ULPI</li> <li>■ 2'b11: UTMI+ and ULPI</li> </ul> <p><b>Note:</b> This field is configured using the OTG_HSPHY_INTERFACE parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOHS): High-Speed interface not supported</li> <li>■ 0x1 (UTMIPLUS): High Speed Interface UTMI+ is supported</li> <li>■ 0x2 (ULPI): High Speed Interface ULPI is supported</li> <li>■ 0x3 (UTMIPUSULPI): High Speed Interfaces UTMI+ and ULPI is supported</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> Always</p>

**Table 7-24 Fields for Register: GHWCFG2 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
5	SingPnt	R	<p>Point-to-Point (SingPnt)</p> <ul style="list-style-type: none"> <li>■ 1'b0: Multi-point application (hub and split support)</li> <li>■ 1'b1: Single-point application (no hub and split support)</li> </ul> <p><b>Note:</b> This field is configured using the OTG_SINGLE_POINT parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MULTIPOINT): Multi-point application (hub and split support)</li> <li>■ 0x1 (SINGLEPOINT): Single-point application (no hub and split support)</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> Always</p>
4:3	OtgArch	R	<p>Architecture (OtgArch)</p> <ul style="list-style-type: none"> <li>■ 2'b00: Slave-Only</li> <li>■ 2'b01: External DMA</li> <li>■ 2'b10: Internal DMA</li> <li>■ Others: Reserved</li> </ul> <p><b>Note:</b> This field is configured using the OTG_ARCHITECTURE parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (SLAVEMODE): Slave Mode</li> <li>■ 0x1 (EXTERNALDMA): External DMA Mode</li> <li>■ 0x2 (INTERNALDMA): Internal DMA Mode</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> Always</p>

**Table 7-24 Fields for Register: GHWCFG2 (Continued)**

Bits	Name	Memory Access	Description
2:0	OtgMode	R	<p>Mode of Operation (OtgMode)</p> <ul style="list-style-type: none"> <li>■ 3'b000: HNP- and SRP-Capable OTG (Host &amp; Device)</li> <li>■ 3'b001: SRP-Capable OTG (Host &amp; Device)</li> <li>■ 3'b010: Non-HNP and Non-SRP Capable OTG (Host and Device)</li> <li>■ 3'b011: SRP-Capable Device</li> <li>■ 3'b100: Non-OTG Device</li> <li>■ 3'b101: SRP-Capable Host</li> <li>■ 3'b110: Non-OTG Host</li> <li>■ Others: Reserved</li> </ul> <p><b>Note:</b> This field is configured using the OTG_MODE parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (HNPSRP): HNP- and SRP-Capable OTG (Host and Device)</li> <li>■ 0x1 (SRPOTG): SRP-Capable OTG (Host and Device)</li> <li>■ 0x2 (NHNPNCSR): Non-HNP and Non-SRP Capable OTG (Host and Device)</li> <li>■ 0x3 (SRPCAPD): SRP-Capable Device</li> <li>■ 0x4 (NONOTGD): Non-OTG Device</li> <li>■ 0x5 (SRPCAPH): SRP-Capable Host</li> <li>■ 0x6 (NONOTGH): Non-OTG Host</li> </ul> <p><b>Value After Reset:</b> * Varies based on Configuration</p> <p><b>Exists:</b> Always</p>

### 7.1.20 GHWCFG3

- **Name:** User Hardware Configuration 3 Register
- **Description:** This register contains configuration options selected using coreConsultant.
- **Size:** 32 bits
- **Offset:** 0x4c
- **Exists:** Always

DfifoDepth	31:16	15	14	13	12	11	10	9	8	7	6:4	3:0
LPMMode												
BCSSupport												
HSICMode												
ADPSupport												
RstType												
OptFeature												
VndctSupt												
I2CIntSel												
OtgEn												
PktSizeWidth												
XferSizeWidth												

**Table 7-25 Fields for Register: GHWCFG3**

Bits	Name	Memory Access	Description
31:16	DfifoDepth	R	<p>DFIFO Depth (DfifoDepth - EP_LOC_CNT) This value is in terms of 32-bit words.</p> <ul style="list-style-type: none"> <li>■ Minimum value is 32</li> <li>■ Maximum value is 32,768</li> </ul> <p><b>Note:</b> This field is configured using the OTG_DFIFO_DEPTH parameter. For more information on EP_LOC_CNT, see the "Endpoint Information Controller (EPINFO_CTL)" section.</p> <p><b>Value After Reset:</b> 0x400</p> <p><b>Exists:</b> Always</p>
15	LPMMode	R	<p>LPM mode specified for Mode of Operation.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): LPM disabled</li> <li>■ 0x1 (ENABLED): LPM enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-25 Fields for Register: GHWCFG3 (Continued)**

Bits	Name	Memory Access	Description
14	BCSupport	R	<p>This bit indicates the controller support for Battery Charger.</p> <ul style="list-style-type: none"> <li>■ 0 - No Battery Charger Support</li> <li>■ 1 - Battery Charger support present</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No Battery Charger Support</li> <li>■ 0x1 (ENABLED): Battery Charger Support present</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
13	HSICMode	R	<p>HSIC mode specified for Mode of Operation</p> <p>Value Range: 0 - 1</p> <ul style="list-style-type: none"> <li>■ 1: HSIC-capable with shared UTMI PHY interface</li> <li>■ 0: Non-HSIC-capable</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No HSIC capability</li> <li>■ 0x1 (ENABLED): HSIC-capable with shared UTMI PHY interface</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
12	ADPSupport	R	<p>This bit indicates whether ADP logic is present within or external to the controller</p> <ul style="list-style-type: none"> <li>■ 0: No ADP logic present with the controller</li> <li>■ 1: ADP logic is present along with the controller.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): ADP logic is not present along with the controller</li> <li>■ 0x1 (ENABLED): ADP logic is present along with the controller</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-25 Fields for Register: GHWCFG3 (Continued)**

Bits	Name	Memory Access	Description
11	RstType	R	<p>Reset Style for Clocked always Blocks in RTL (RstType)</p> <ul style="list-style-type: none"> <li>■ 1'b0: Asynchronous reset is used in the controller</li> <li>■ 1'b1: Synchronous reset is used in the controller</li> </ul> <p><b>Note:</b> This field is configured using the OTG_SYNC_RESET_TYPE parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ASYNCRST): Asynchronous reset is used in the core</li> <li>■ 0x1 (SYNCRST): Synchronous reset is used in the core</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
10	OptFeature	R	<p>Optional Features Removed (OptFeature)</p> <p>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.</p> <ul style="list-style-type: none"> <li>■ 1'b0: No</li> <li>■ 1'b1: Yes</li> </ul> <p><b>Note:</b> This field is configured using the OTG_RM_OPT_FEATURES parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Optional features were not Removed</li> <li>■ 0x1 (ENABLED): Optional Features have been Removed</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p>
9	VndctlSupt	R	<p>Vendor Control Interface Support (VndctlSupt)</p> <ul style="list-style-type: none"> <li>■ 1'b0: Vendor Control Interface is not available on the core.</li> <li>■ 1'b1: Vendor Control Interface is available.</li> </ul> <p><b>Note:</b> This field is configured using the OTG_VENDOR_CTL_INTERFACE parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Vendor Control Interface is not available.</li> <li>■ 0x1 (ENABLED): Vendor Control Interface is available.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-25 Fields for Register: GHWCFG3 (Continued)**

Bits	Name	Memory Access	Description
8	I2CIntSel	R	<p>I2C Selection (I2CIntSel)</p> <ul style="list-style-type: none"> <li>■ 1'b0: I2C Interface is not available on the controller.</li> <li>■ 1'b1: I2C Interface is available on the controller.</li> </ul> <p><b>Note:</b> This field is configured using the OTG_I2C_INTERFACE parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): I2C Interface is not available</li> <li>■ 0x1 (ENABLED): I2C Interface is available</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
7	OtgEn	R	<p>OTG Function Enabled (OtgEn)</p> <p>The application uses this bit to indicate the OTG capabilities of the controller .</p> <ul style="list-style-type: none"> <li>■ 1'b0: Not OTG capable</li> <li>■ 1'b1: OTG Capable</li> </ul> <p><b>Note:</b> This field is configured using the OTG_MODE parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Not OTG Capable</li> <li>■ 0x1 (ENABLED): OTG Capable</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p>

**Table 7-25 Fields for Register: GHWCFG3 (Continued)**

Bits	Name	Memory Access	Description
6:4	PktSizeWidth	R	<p>Width of Packet Size Counters (PktSizeWidth)</p> <ul style="list-style-type: none"> <li>■ 3'b000: 4 bits</li> <li>■ 3'b001: 5 bits</li> <li>■ 3'b010: 6 bits</li> <li>■ 3'b011: 7 bits</li> <li>■ 3'b100: 8 bits</li> <li>■ 3'b101: 9 bits</li> <li>■ 3'b110: 10 bits</li> <li>■ Others: Reserved</li> </ul> <p><b>Note:</b> This field is configured using the OTG_PACKET_COUNT_WIDTH parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (BITS4): Width of Packet Size Counter 4</li> <li>■ 0x1 (BITS5): Width of Packet Size Counter 5</li> <li>■ 0x2 (BITS6): Width of Packet Size Counter 6</li> <li>■ 0x3 (BITS7): Width of Packet Size Counter 7</li> <li>■ 0x4 (BITS8): Width of Packet Size Counter 8</li> <li>■ 0x5 (BITS9): Width of Packet Size Counter 9</li> <li>■ 0x6 (BITS10): Width of Packet Size Counter 10</li> </ul> <p><b>Value After Reset:</b> 0x6</p> <p><b>Exists:</b> Always</p>

**Table 7-25 Fields for Register: GHWCFG3 (Continued)**

Bits	Name	Memory Access	Description
3:0	XferSizeWidth	R	<p>Width of Transfer Size Counters (XferSizeWidth)</p> <ul style="list-style-type: none"> <li>■ 4'b0000: 11 bits</li> <li>■ 4'b0001: 12 bits</li> <li>...</li> <li>■ 4'b1000: 19 bits</li> <li>■ Others: Reserved</li> </ul> <p><b>Note:</b> This field is configured using the OTG_PACKET_COUNT_WIDTH parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (WIDTH11): Width of Transfer Size Counter 11 bits</li> <li>■ 0x1 (WIDTH12): Width of Transfer Size Counter 12 bits</li> <li>■ 0x2 (WIDTH13): Width of Transfer Size Counter 13 bits</li> <li>■ 0x3 (WIDTH14): Width of Transfer Size Counter 14 bits</li> <li>■ 0x4 (WIDTH15): Width of Transfer Size Counter 15 bits</li> <li>■ 0x5 (WIDTH16): Width of Transfer Size Counter 16 bits</li> <li>■ 0x6 (WIDTH17): Width of Transfer Size Counter 17 bits</li> <li>■ 0x7 (WIDTH18): Width of Transfer Size Counter 18 bits</li> <li>■ 0x8 (WIDTH19): Width of Transfer Size Counter 19 bits</li> </ul> <p><b>Value After Reset:</b> 0x8</p> <p><b>Exists:</b> Always</p>

### 7.1.21 GHWCFG4

- **Name:** User Hardware Configuration 4 Register
- **Description:** This register contains configuration options selected using coreConsultant.
- Note:** Bit [31] is available only when Scatter/Gather DMA mode is enabled. When Scatter/Gather DMA mode is disabled, this field is reserved.
- **Size:** 32 bits
- **Offset:** 0x50
- **Exists:** Always

31	DescDMA	30	DescDMAEnabled	30	29:26	INEps	25	DedFifoMode	25	SessEndFltr	24	BValidFltr	23	AValidFltr	22	VBusValidFltr	21	IddgFltr	20	NumCtlEps	19:16	PhyDataWidth	15:14	EnhancedLPMSupt	13	ACGSupt	12	ipgisocSupt	11	ServIntFlow	10	EnhancedLPMSupt1	9	Reserved_8	8	ExtendedHibernation	7	Hibernation	6	AhbFreq	5	PartialPwrDn	4	NumDevPerioEps	3:0
----	---------	----	----------------	----	-------	-------	----	-------------	----	-------------	----	------------	----	------------	----	---------------	----	----------	----	-----------	-------	--------------	-------	-----------------	----	---------	----	-------------	----	-------------	----	------------------	---	------------	---	---------------------	---	-------------	---	---------	---	--------------	---	----------------	-----

Table 7-26 Fields for Register: GHWCFG4

Bits	Name	Memory Access	Description
31	DescDMA	R	<p>Scatter/Gather DMA configuration</p> <ul style="list-style-type: none"> <li>■ 1'b0: Non Dynamic configuration</li> <li>■ 1'b1: Dynamic configuration</li> </ul> <p><b>Note:</b> This field is configured using the OTG_EN_DESC_DMA parameter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (CONFIG1): Non Dynamic configuration</li> <li>■ 0x1 (CONFIG2): Dynamic configuration</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-26 Fields for Register: GHWCFG4 (Continued)**

Bits	Name	Memory Access	Description
30	DescDMAEnabled	R	<p>Scatter/Gather DMA configuration</p> <ul style="list-style-type: none"> <li>■ 1'b0: Non-Scatter/Gather DMA configuration</li> <li>■ 1'b1: Scatter/Gather DMA configuration</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Non-Scatter/Gather DMA configuration</li> <li>■ 0x1 (ENABLE): Scatter/Gather DMA configuration</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
29:26	INEps	R	<p>Number of Device Mode IN Endpoints Including Control Endpoints (INEps)</p> <ul style="list-style-type: none"> <li>■ 0: 1 IN Endpoint</li> <li>■ 1: 2 IN Endpoints</li> <li>....</li> <li>■ 15: 16 IN Endpoints</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ENDPT1): 1 IN Endpoint</li> <li>■ 0x1 (ENDPT2): 2 IN Endpoints</li> <li>■ 0x2 (ENDPT3): 3 IN Endpoints</li> <li>■ 0x3 (ENDPT4): 4 IN Endpoints</li> <li>■ 0x4 (ENDPT5): 5 IN Endpoints</li> <li>■ 0x5 (ENDPT6): 6 IN Endpoints</li> <li>■ 0x6 (ENDPT7): 7 IN Endpoints</li> <li>■ 0x7 (ENDPT8): 8 IN Endpoints</li> <li>■ 0x8 (ENDPT9): 9 IN Endpoints</li> <li>■ 0x9 (ENDPT10): 10 IN Endpoints</li> <li>■ 0xa (ENDPT11): 11 IN Endpoints</li> <li>■ 0xb (ENDPT12): 12 IN Endpoints</li> <li>■ 0xc (ENDPT13): 13 IN Endpoints</li> <li>■ 0xd (ENDPT14): 14 IN Endpoints</li> <li>■ 0xe (ENDPT15): 15 IN Endpoints</li> <li>■ 0xf (ENDPT16): 16 IN Endpoints</li> </ul> <p><b>Value After Reset:</b> 0x2</p> <p><b>Exists:</b> Always</p>

**Table 7-26 Fields for Register: GHWCFG4 (Continued)**

Bits	Name	Memory Access	Description
25	DedFifoMode	R	<p>Enable Dedicated Transmit FIFO for device IN Endpoints (DedFifoMode)</p> <ul style="list-style-type: none"> <li>■ 1'b0 : Dedicated Transmit FIFO Operation not enabled.</li> <li>■ 1'b1 : Dedicated Transmit FIFO Operation enabled.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Dedicated Transmit FIFO Operation not enabled</li> <li>■ 0x1 (ENABLED): Dedicated Transmit FIFO Operation enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
24	SessEndFltr	R	<p>session_end Filter Enabled (SessEndFltr)</p> <ul style="list-style-type: none"> <li>■ 1'b0: No filter</li> <li>■ 1'b1: Filter</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No filter</li> <li>■ 0x1 (ENABLED): Filter</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
23	BValidFltr	R	<p>b_valid Filter Enabled (BValidFltr)</p> <ul style="list-style-type: none"> <li>■ 1'b0: No filter</li> <li>■ 1'b1: Filter</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No Filter</li> <li>■ 0x1 (ENABLED): Filter</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-26 Fields for Register: GHWCFG4 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
22	AValidFltr	R	<p>a_valid Filter Enabled (AValidFltr)</p> <ul style="list-style-type: none"> <li>■ 1'b0: No filter</li> <li>■ 1'b1: Filter</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No filter</li> <li>■ 0x1 (ENABLED): Filter</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
21	VBusValidFltr	R	<p>VBUS Valid Filter Enabled (VBusValidFltr)</p> <ul style="list-style-type: none"> <li>■ 1'b0: No filter</li> <li>■ 1'b1: Filter</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Vbus Valid Filter Disabled</li> <li>■ 0x1 (ENABLED): Vbus Valid Filter Enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
20	IddgFltr	R	<p>IDDG Filter Enable (IddgFltr)</p> <ul style="list-style-type: none"> <li>■ 1'b0: No filter</li> <li>■ 1'b1: Filter</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Iddig Filter Disabled</li> <li>■ 0x1 (ENABLED): Iddig Filter Enabled</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p>

**Table 7-26 Fields for Register: GHWCFG4 (Continued)**

Bits	Name	Memory Access	Description
19:16	NumCtlEps	R	<p>Number of Device Mode Control Endpoints in Addition to Endpoint 0 (NumCtlEps) Range: 0-15</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ENDPT0): End point 0</li> <li>■ 0x1 (ENDPT1): End point 1</li> <li>■ 0x2 (ENDPT2): End point 2</li> <li>■ 0x3 (ENDPT3): End point 3</li> <li>■ 0x4 (ENDPT4): End point 4</li> <li>■ 0x5 (ENDPT5): End point 5</li> <li>■ 0x6 (ENDPT6): End point 6</li> <li>■ 0x7 (ENDPT7): End point 7</li> <li>■ 0x8 (ENDPT8): End point 8</li> <li>■ 0x9 (ENDPT9): End point 9</li> <li>■ 0xa (ENDPT10): End point 10</li> <li>■ 0xb (ENDPT11): End point 11</li> <li>■ 0xc (ENDPT12): End point 12</li> <li>■ 0xd (ENDPT13): End point 13</li> <li>■ 0xe (ENDPT14): End point 14</li> <li>■ 0xf (ENDPT15): End point 15</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
15:14	PhyDataWidth	R	<p>UTMI+ PHY/ULPI-to-Internal UTMI+ Wrapper Data Width (PhyDataWidth)&lt;vr&gt;When a ULPI PHY is used, an internal wrapper converts ULPI to UTMI+.</p> <p>■ 2'b00: 8 bits</p> <p>■ 2'b01: 16 bits</p> <p>■ 2'b10: 8/16 bits, software selectable</p> <p>■ Others: Reserved</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (WIDTH1): 8 bits</li> <li>■ 0x1 (WIDTH2): 16 bits</li> <li>■ 0x2 (WIDTH3): 8/16 bits, software selectable</li> </ul> <p><b>Value After Reset:</b> 0x2</p> <p><b>Exists:</b> Always</p>

**Table 7-26 Fields for Register: GHWCFG4 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
13	EnhancedLPMSupt	R	<p>Enhanced LPM Support (EnhancedLPMSupt)  This bit indicates that the controller supports the following behavior:  L1 Entry Behavior based on FIFO Status</p> <ul style="list-style-type: none"> <li>■ TX FIFO</li> <li>■ Accept L1 Request even if ISOC IN TX FIFO is not empty.</li> <li>■ Reject L1 Request if Non-Periodic TX FIFO is not empty.</li> <li>■ Ensure application can flush the TX FIFO while the Controller is in L1.</li> <li>■ RX FIFO</li> <li>■ Accept L1 Request even if RX FIFO (common to Periodic and Non-Periodic) is not empty.</li> <li>■ Accept L1 Request but delay SLEEP assertion until RX SINK Buffer is empty.</li> </ul> <p>Prevent L1 Entry if a Control Transfer is in progress on any Control Endpoint. Ability to Flush TxFIFO even if PHY Clock is gated.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (ENABLED): Enhanced LPM Support is enabled</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p>
12	ACGSupt	R	<p>Active Clock Gating Support  This bit indicates that the controller supports the Dynamic (Switching) Power Reduction during periods when there is no USB and AHB Traffic.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Active Clock Gating is not enabled.</li> <li>■ 1'b1: Active Clock Gating Enabled.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED)</li> <li>■ 0x1 (ENABLED): Active Clock Gating Support</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-26 Fields for Register: GHWCFG4 (Continued)**

Bits	Name	Memory Access	Description
11	ipgisocSupt	R	<p>Interpacket Gap ISOC OUT Worst-case Support (ipgisocSupt)  This bit indicates that the controller supports the worst-case scenario of Rx followed by Rx Inter Packet Gap (IPG) (32-bit times) as per the UTMI Specification for any token following an ISOC OUT token. Without this support, when any token follows an ISOC OUT token with the worst-case IPG, the controller will not detect the followed token. The worst-case IPG of the controller without this support depends on the AHB and PHY clock frequency. By default IPG Support is enabled.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Interpacket Gap ISOC OUT Worst-case Support is Disabled</li> <li>■ 0x1 (ENABLED): Interpacket Gap ISOC OUT Worst-case Support is Enabled (Default)</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>
10	ServIntFlow	R	<p>Service Interval Flow  This bit indicates that the controller supports Service-Interval based scheduling flow for ISOC IN EPs.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Service Interval Flow not supported</li> <li>■ 0x1 (ENABLED): Service Interval Flow supported</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>
9	EnhancedLPMSupt1	R	<p>Enhanced LPM Support1 (EnhancedLPMSupt1)</p> <ul style="list-style-type: none"> <li>■ This bit indicates that the controller supports L1 entry based on FIFO status.</li> <li>■ Accept L1 Request even if Bulk/Interrupt TxFIFO is not empty.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Reject L1 Request even if Non-Periodic (Bulk/Interrupt) TxFIFO is not empty.</li> <li>■ 0x1 (ENABLED): Accept L1 Request even if Non-Periodic (Bulk/Interrupt) TxFIFO is not empty</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>

**Table 7-26 Fields for Register: GHWCFG4 (Continued)**

Bits	Name	Memory Access	Description
8	Reserved_8	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> untestable</p> <p><b>Write Constraint:</b> writeAsRead</p>
7	ExtendedHibernation	R	<p>Enable Hibernation</p> <ul style="list-style-type: none"> <li>■ 1'b0: Extended Hibernation feature not enabled</li> <li>■ 1'b1: Extended Hibernation feature enabled</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Extended Hibernation feature not enabled</li> <li>■ 0x1 (ENABLED): Extended Hibernation feature enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
6	Hibernation	R	<p>Enable Hibernation (Hibernation)</p> <ul style="list-style-type: none"> <li>■ 1'b0: Hibernation feature not enabled</li> <li>■ 1'b1: Hibernation feature enabled</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Hibernation feature disabled</li> <li>■ 0x1 (ENABLED): Hibernation feature enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
5	AhbFreq	R	<p>Minimum AHB Frequency Less Than 60 MHz (AhbFreq)</p> <ul style="list-style-type: none"> <li>■ 1'b0: No</li> <li>■ 1'b1: Yes</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Minimum AHB Frequency More Than 60 MHz</li> <li>■ 0x1 (ENABLED): Minimum AHB Frequency Less Than 60 MHz</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p>

**Table 7-26 Fields for Register: GHWCFG4 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
4	PartialPwrDn	R	<p>Enable Partial Power Down (PartialPwrDn)</p> <ul style="list-style-type: none"> <li>■ 1'b0: Partial Power Down Not Enabled</li> <li>■ 1'b1: Partial Power Down Enabled</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Partial Power Down disabled</li> <li>■ 0x1 (ENABLED): Partial Power Down enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
3:0	NumDevPerioEps	R	<p>Number of Device Mode Periodic IN Endpoints (NumDevPerioEps) Range: 0-15</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (Value0): Number of Periodic IN EPs is 0</li> <li>■ 0x1 (Value1): Number of Periodic IN EPs is 1</li> <li>■ 0x2 (Value2): Number of Periodic IN EPs is 2</li> <li>■ 0x3 (Value3): Number of Periodic IN EPs is 3</li> <li>■ 0x4 (Value4): Number of Periodic IN EPs is 4</li> <li>■ 0x5 (Value5): Number of Periodic IN EPs is 5</li> <li>■ 0x6 (Value6): Number of Periodic IN EPs is 6</li> <li>■ 0x7 (Value7): Number of Periodic IN EPs is 7</li> <li>■ 0x8 (Value8): Number of Periodic IN EPs is 8</li> <li>■ 0x9 (Value9): Number of Periodic IN EPs is 9</li> <li>■ 0xa (Value10): Number of Periodic IN EPs is 10</li> <li>■ 0xb (Value11): Number of Periodic IN EPs is 11</li> <li>■ 0xc (Value12): Number of Periodic IN EPs is 12</li> <li>■ 0xd (Value13): Number of Periodic IN EPs is 13</li> <li>■ 0xe (Value14): Number of Periodic IN EPs is 14</li> <li>■ 0xf (Value15): Number of Periodic IN EPs is 15</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.22 GLPMCFG

- **Name:** LPM Config Register
- **Description:** Register to control the LPM functionality of the controller.
- **Size:** 32 bits
- **Offset:** 0x54
- **Exists:** OTG\_ENABLE\_LPM==1 || OTG\_ENABLE\_HSIC==1

31	InvSelHsic	30	HSICCon	29	LPM_RestoreSlpSts	28	LPM_EnBESL	27:25	LPM_RetryCnt_Sts	SndLPM	24	LPM_Retry_Cnt	23:21	LPM_Chnl_Idx	20:17	L1ResumeOK	16	SlpSts	15	CoreL1Res	14:13	HIRD_Thres	12:8	EnbSlpM	7	bRemoteWake	6	HIRD	5:2	AppL1Ress	1	LPMCap	0
----	------------	----	---------	----	-------------------	----	------------	-------	------------------	--------	----	---------------	-------	--------------	-------	------------	----	--------	----	-----------	-------	------------	------	---------	---	-------------	---	------	-----	-----------	---	--------	---

Table 7-27 Fields for Register: GLPMCFG

Bits	Name	Memory Access	Description
31	InvSelHsic	OTG_ENABLE_HSIC ? R/W : R	<p>HSIC-Invert Select HSIC (InvSelHsic)  The application uses this bit to control the 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 then connect to only PHYs which are not HSIC capable.  If the core operates as HSIC-capable, it can then connect to only PHYs which are HSIC capable.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (InvSelHsic_0): HSIC capability is enabled if if_sel_hsic=1 and disabled if if_sel_hsic=0</li> <li>■ 0x1 (InvSelHsic_1): HSIC capability is disabled if if_sel_hsic=1 and enabled if if_sel_hsic=0</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_ENABLE_HSIC==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
30	HSICCon	OTG_EN ABLE_H SIC== 1 ? R/W : R	<p>HSIC-Connect (HSICCon)  The application must use this bit to initiate HSIC ATTACH sequence.</p> <p>Host Mode: Once this bit is set, the Host Core configures to drive HSIC IDLE state (STROBE=1&amp;DATA=0) on the BUS. It then waits for device to initiate CONNECT.</p> <p>Device Mode: Once this bit is set, the Device Core waits for HSIC IDLE Linestate on the BUS. Upon receiving the IDLE linestate it then initiates HSIC CONNECT.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE_HSIC_CONNECT): In HOST/DEVICE mode, core does not initiate HSIC connect</li> <li>■ 0x1 (ENABLE_HSIC_CONNECT): In HOST/DEVICE mode, core initiates the HSIC connect after HSIC IDLE linestate</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_HSIC==1</p>
29	LPM_RestoreSlpSts	R/W	<p>LPM Restore Sleep Status (LPM_RestoreSlpSts)</p> <p>When the application power gates the core (Partial Power Down / Hibernation) the application needs to program this bit to restore the LPM status in the core.</p> <p>The application needs to program this bit, during restore process, based on whether it had decided to go into Shallow Sleep (Clock Gating Only) or Deep Sleep (Power Gating) based on the BESL value received from the Host</p> <ul style="list-style-type: none"> <li>■ 1'b0: The application puts the core in Shallow Sleep based on BESL value from the Host</li> <li>■ 1'b1: The application puts the core in Deep Sleep based on BESL value from the Host</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Puts the core in Shallow Sleep mode based on the BESL value from the Host</li> <li>■ 0x1 (ENABLED): Puts the core in Deep Sleep mode based on the BESL value from the Host</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 5 &amp;&amp; OTG_MODE != 6) &amp;&amp; OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
28	LPM_EnBESL	R/W	<p>LPM Enable BESL (LPM_EnBESL)  This bit enables the BESL feature as defined in LPM Errata</p> <ul style="list-style-type: none"> <li>■ 1'b0: The core works as per USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification as of July 16, 2007</li> <li>■ 1'b1: The core works as per the LPM Errata</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): BESL is disabled</li> <li>■ 0x1 (ENABLED): BESL is enabled as defined in LPM Errata</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM==1</p>
27:25	LPM_RetryCnt_Sts	R	<p>LPM Retry Count Status (LPM_RetryCnt_Sts)  <b>Host Mode:</b> Number of LPM Host Retries still remaining to be transmitted for the current LPM sequence.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (RETRY_Rem0): Zero LPM retries remaining</li> <li>■ 0x1 (RETRY_Rem1): One LPM retry remaining</li> <li>■ 0x2 (RETRY_Rem2): Two LPM retries remaining</li> <li>■ 0x3 (RETRY_Rem3): Three LPM retries remaining</li> <li>■ 0x4 (RETRY_Rem4): Four LPM retries remaining</li> <li>■ 0x5 (RETRY_Rem5): Five LPM retries remaining</li> <li>■ 0x6 (RETRY_Rem6): Six LPM retries remaining</li> <li>■ 0x7 (RETRY_Rem7): Seven LPM retries remaining</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
24	SndLPM	R/W	<p>Send LPM Transaction (SndLPM)            Host Mode: When set by application software, an LPM transaction containing two tokens, EXT &amp; LPM is sent. Cleared by the hardware once a valid response (STALL./NYET/ACK) is received from the Device or the core had finished transmitting the programmed number of LPM retries. Note that this bit should be set only when the host is connected to local port.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): In host-only mode: Received the response from the device for the LPM transaction</li> <li>■ 0x1 (ENABLED): In host-only mode: Sending LPM transaction containing EXT and LPM tokens</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4) &amp;&amp; OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
23:21	LPM_Retry_Cnt	R/W	<p>LPM Retry Count (LPM_Retry_Cnt)</p> <p><b>Host Mode:</b> Number of additional LPM retries that the HOST would perform if the Device Response was an ERROR until a valid device response is received (STALL/NYET/ACK).</p> <p><b>Device Mode:</b> LPM Accept Control (LPM_Accept_Ctrl)</p> <p>LPM_Accept_Ctrl[1]: The application can use this bit to reject an LPM token (NYET) between multiple stages of a single control transfer</p> <ul style="list-style-type: none"> <li>■ 1'b0: Accept(ACK) LPM token during Setup, Data and Status stage of control transfer.</li> <li>■ 1'b1: Reject(NYET) LPM token during Setup, Data and Status stage of control transfer.</li> </ul> <p>LPM_Accept_Ctrl[2]: The application can use this bit to accept an LPM token even if data is present in the ISOC endpoint TxFIFO. This bit is applicable only for Dedicated TxFIFO configurations (OTG_EN_DED_TX_FIFO=1).</p> <ul style="list-style-type: none"> <li>■ 1'b0: Reject (NYET) LPM token, when data is present in TxFIFO for ISOC endpoints.</li> <li>■ 1'b1: Accept(ACK) LPM token, when data is present in TxFIFO for ISOC endpoints.</li> </ul> <p>LPM_Accept_Ctrl[3]: The application can use this bit to accept an LPM token even if data is present in the BULK endpoint TxFIFO. This bit is applicable only for Dedicated TxFIFO configurations (OTG_EN_DED_TX_FIFO=1).</p> <ul style="list-style-type: none"> <li>■ 1'b0: Reject (NYET) LPM token, when data is present in TxFIFO for BULK endpoints.</li> <li>■ 1'b1: Accept(ACK) LPM token, when data is present in TxFIFO for BULK endpoints.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (RETRY0): Zero LPM retries</li> <li>■ 0x1 (RETRY1): One LPM retry</li> <li>■ 0x2 (RETRY2): Two LPM retries</li> <li>■ 0x3 (RETRY3): Three LPM retries</li> <li>■ 0x4 (RETRY4): Four LPM retries</li> <li>■ 0x5 (RETRY5): Five LPM retries</li> <li>■ 0x6 (RETRY6): Six LPM retries</li> <li>■ 0x7 (RETRY7): Seven LPM retries</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
20:17	LPM_Chnl_Idx	R/W	<p>LPM Channel Index</p> <p><b>Host Mode:</b> The channel number on which the LPM transaction has to 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 end point number programmed in the corresponding channel into the LPM transaction.</p> <p><b>Device Mode:</b> LPM Accept Control (LPM_Accept_Ctrl) LPM_Accept_Ctrl[0] (LPM_Chnl_Idx[3]): The application can use this bit to accept an LPM token even if data is present in the INTR endpoint TxFIFO. This bit is applicable only for Dedicated TX FIFO configurations (OTG_EN_DED_TX_FIFO=1).</p> <ul style="list-style-type: none"> <li>■ 1'b0:Reject (NYET) LPM token, when data is present in txfifo for INTR endpoints.</li> <li>■ 1'b1:Accept(ACK) LPM token, when data is present in txfifo for INTR endpoints.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (CH0): Channel 0</li> <li>■ 0x1 (CH1): Channel 1</li> <li>■ 0x2 (CH2): Channel 2</li> <li>■ 0x3 (CH3): Channel 3</li> <li>■ 0x4 (CH4): Channel 4</li> <li>■ 0x5 (CH5): Channel 5</li> <li>■ 0x6 (CH6): Channel 6</li> <li>■ 0x7 (CH7): Channel 7</li> <li>■ 0x8 (CH8): Channel 8</li> <li>■ 0x9 (CH9): Channel 9</li> <li>■ 0xa (CH10): Channel 10</li> <li>■ 0xb (CH11): Channel 11</li> <li>■ 0xc (CH12): Channel 12</li> <li>■ 0xd (CH13): Channel 13</li> <li>■ 0xe (CH14): Channel 14</li> <li>■ 0xf (CH15): Channel15</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE != 3    OTG_MODE != 4) &amp;&amp; OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
16	L1ResumeOK	R	<p>Sleep State Resume OK (L1ResumeOK)</p> <p><b>Device and Host Mode:</b></p> <p>Indicates that the application or host can start resume from Sleep state. This bit is valid in LPM sleep (L1) state. It is set in sleep mode after a delay of 50 micro sec (TL1Residency). The bit is reset when SlpSts is 0.</p> <ul style="list-style-type: none"> <li>■ 1'b0: The application/core cannot start resume from Sleep state.</li> <li>■ 1'b1: The application/core can start resume from Sleep state.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTOK): The application/core cannot start Resume from Sleep state</li> <li>■ 0x1 (OK): The application/core can start Resume from Sleep state</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
15	SlpSts	R	<p>Port Sleep Status (SlpSts)  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 TL1TokenRetry timer has expired. To stop the PHY clock, the application must set the Port Clock Stop bit, which asserts the PHY Suspend input signal. The application must rely on SlpSts and not ACK in CoreL1Res to confirm transition into sleep.  The core comes out of sleep:</p> <ul style="list-style-type: none"> <li>■ When there is any activity on the USB line_state</li> <li>■ 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.</li> </ul> <p>Host Mode: The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device. The read value of this bit reflects the current Sleep status of the port.  The core clears this bit after:</p> <ul style="list-style-type: none"> <li>■ The core detects a remote L1 Wakeup signal;</li> <li>■ The application sets the Port Reset bit or the Port L1Resume bit in the HPRT register; or</li> <li>■ 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).</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (CORE_NOT_IN_L1): In Host or Device mode, this bit indicates core is not in L1</li> <li>■ 0x1 (CORE_IN_L1): In Host mode, this bit indicates the core transitions to Sleep state as a successful LPM transaction. In Device mode, the core enters the Sleep state when an ACK response is sent to an LPM transaction</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
14:13	CoreL1Res	R	<p>LPM Response (CoreL1Res)  Device Mode: The response of the core to LPM transaction received is reflected in these two bits.  Host Mode: Handshake response received from local device for LPM transaction</p> <ul style="list-style-type: none"> <li>■ 11 - ACK</li> <li>■ 10 - NYET</li> <li>■ 01 - STALL</li> <li>■ 00 - ERROR (No handshake response)</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (LPMRESP1): ERROR : No handshake response</li> <li>■ 0x1 (LPMRESP2): STALL response</li> <li>■ 0x2 (LPMRESP3): NYET response</li> <li>■ 0x3 (LPMRESP4): ACK response</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
12:8	HIRD_Thres	R/W	<p>BESL/HIRD Threshold (HIRD_Thres)</p> <p><b>Device Mode:</b></p> <p><b>Note:</b> When a ULPI/UTMI PHY associated with the controller (Host or device) does not support the LPM sleep feature clock gating, it is recommended to keep the HIRD_Thres[12] bit in disabled state (= 1'b0). This ensures the ULPI wrapper is in enabled mode and detects any wakeup events.</p> <ul style="list-style-type: none"> <li>■ EnBESL = 1'b0: The core puts the PHY into deep low power mode in L1 (by core asserting L1SuspendM) when HIRD value is greater than or equal to the value defined in this field HIRD_Thres[3:0] and HIRD_Thres[4] is set to 1b1.</li> <li>■ EnBESL = 1'b1: The core puts the PHY into deep low power mode in L1 (by core asserting L1SuspendM) when BESL value is greater than or equal to the value defined in this field BESL_Thres[3:0] and BESL_Thres [4] is set to 1'b1.</li> <li>■ DCTL.DeepSleepBESLReject = 1'b1: In device initiated resume, the core expects the Host to resume service to the device within the BESL value corresponding to L1 exit time specified in HIRD_Thres[3:0]. The Device sends a NYET response when the received HIRD in LPM token is greater than HIRD threshold.</li> </ul> <p><b>Host Mode:</b> The core puts the PHY into deep low power mode in L1 (by core asserting L1SuspendM) 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 host (TL1HubDrvResume2) on the USB bus when it detects device initiated resume.</p> <p>To differentiate between Deep Sleep and Shallow Sleep, HIRD greater than or equal to HIRD threshold comparison is done. For differentiating between NYET or ACK response for LPM token HIRD greater than HIRD Threshold comparison is used.</p> <ul style="list-style-type: none"> <li>■ When EnBESL = 1'b0, HIRD_Thres should not be programmed with a value greater than 4'b1100 in host mode since this will exceed maximum TL1HubDrvResume2.</li> <li>■ When EnBESL = 1'b1, HIRD_Thres should not be programmed with a value greater than 4'b0110 in host mode since this will exceed maximum TL1HubDrvResume2.</li> </ul> <p>For additional details, see the "Additional Details on GLPMCFG.HIRD_Thres Register Field" section in the databook.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
7	EnbISlpM	R/W	<p>Enable utmi_sleep_n (EnbISlpM)</p> <p><b>Mode:</b> Host and Device</p> <p>For ULPI interface: The application uses this bit to control the utmi_sleep_n assertion to the PHY when in L1 state. For the host, this bit is valid only in Local Device mode.</p> <ul style="list-style-type: none"> <li>■ 1'b0: utmi_sleep_n assertion from the core is not transferred to the external PHY.</li> <li>■ 1'b1: utmi_sleep_n assertion from the core is transferred to the external PHY.</li> </ul> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>■ When a ULPI interface is configured, enabling this bit results in a write to Bit 7 of the ULPI Function Control register. The Synopsys ULPI PHY supports writing to this bit, and in the L1 state asserts SleepM when utmi_l1_suspend_n cannot be asserted.</li> <li>■ When a ULPI/UTMI PHY associated with the controller (Host or device) does not support the LPM sleep feature clock gating, it is recommended to keep the EnbISlpM bit in disabled state (= 1'b0). This ensures the ULPI wrapper is in enabled mode and detects any wakeup events.</li> </ul> <p>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> <ul style="list-style-type: none"> <li>■ 1'b0: utmi_sleep_n assertion from the core is not transferred to the external PHY.</li> <li>■ 1'b1: utmi_sleep_n assertion from the core is transferred to the external PHY when utmi_l1_suspend_n cannot be asserted.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): utmi_sleep_n assertion from the core is not transferred to the external PHY</li> <li>■ 0x1 (ENABLED): utmi_sleep_n assertion from the core is transferred to the external PHY when utmi_l1_suspend_n cannot be asserted</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
6	bRemoteWake	OTG_MO DE!=3 & OTG_MO DE!=4 ? R/W : R	<p>RemoteWakeEnable (bRemoteWake)</p> <p><b>Mode:</b> Host and Device</p> <p>Host Mode: The value of remote wake up to be sent in wlIndex field of LPM transaction.</p> <p>Device Mode: This field is read only. It is updated with the Received LPM Token bRemoteWake bmAttribute when an ACK/NYET/STALL response is sent to an LPM transaction.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Remote Wakeup is disabled</li> <li>■ 0x1 (ENABLED): In Host or device mode, this field takes the value of remote wake up</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
5:2	HIRD	OTG_MO DE!=3 & OTG_MO DE!=4 ? R/W : R	<p>Host-Initiated Resume Duration (HIRD) <b>EnBESL = 1'b0</b> Host Initiated Resume Duration.</p> <ul style="list-style-type: none"> <li>■ 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.</li> <li>■ Device Mode: This field is read only and is updated with the Received LPM Token HIRD bmAttribute when an ACK/NYET/STALL response is sent to an LPM transaction.</li> </ul> <p>If HIRD[3:0],</p> <ul style="list-style-type: none"> <li>■ 4'b0000, THIRD(us) = 50</li> <li>■ 4'b0001, THIRD(us) = 125</li> <li>■ 4'b0010, THIRD(us) = 200</li> <li>■ 4'b0011, THIRD(us) = 275</li> <li>■ 4'b0100, THIRD(us) = 350</li> <li>■ 4'b0101, THIRD(us) = 425</li> <li>■ 4'b0110, THIRD(us) = 500</li> <li>■ 4'b0111, THIRD(us) = 575</li> <li>■ 4'b1000, THIRD(us) = 650</li> <li>■ 4'b1001, THIRD(us) = 725</li> <li>■ 4'b1010, THIRD(us) = 800</li> <li>■ 4'b1011, THIRD(us) = 875</li> <li>■ 4'b1100, THIRD(us) = 950</li> <li>■ 4'b1101, THIRD(us) = 1025</li> <li>■ 4'b1110, THIRD(us) = 1100</li> <li>■ 4'b1111, THIRD(us) = 1175</li> </ul> <p><b>EnBESL = 1'b1</b> Best Effort Service Latency (BESL).</p> <ul style="list-style-type: none"> <li>■ Host Mode: The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration TL1HubDrvResume1 for host-initiated resume.</li> <li>■ Device Mode: This field is updated with the Received LPM Token BESL bmAttribute when an ACK/NYET/STALL response is sent to an LPM transaction.</li> </ul> <p>If BESL[3:0],</p> <ul style="list-style-type: none"> <li>■ 4'b0000, TBESL(us) = 125</li> <li>■ 4'b0001, TBESL(us) = 150</li> <li>■ 4'b0010, TBESL(us) = 200</li> <li>■ 4'b0011, TBESL(us) = 300</li> </ul>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
			<ul style="list-style-type: none"> <li>■ 4'b0100, TBESL(us) = 400</li> <li>■ 4'b0101, TBESL(us) = 500</li> <li>■ 4'b0110, TBESL(us) = 1000</li> <li>■ 4'b0111, TBESL(us) = 2000</li> <li>■ 4'b1000, TBESL(us) = 3000</li> <li>■ 4'b1001, TBESL(us) = 4000</li> <li>■ 4'b1010, TBESL(us) = 5000</li> <li>■ 4'b1011, TBESL(us) = 6000</li> <li>■ 4'b1100, TBESL(us) = 7000</li> <li>■ 4'b1101, TBESL(us) = 8000</li> <li>■ 4'b1110, TBESL(us) = 9000</li> <li>■ 4'b1111, TBESL(us) = 10000</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

Bits	Name	Memory Access	Description
1	AppL1Res	R/W	<p>Mode: Device only LPM response programmed by application (AppL1Res) 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 operates as a non-LPM-capable Device and does not respond to any LPM transactions. If GLPMCFG.LPMCap is 1'b1, the core responds as follows:</p> <ul style="list-style-type: none"> <li>■ 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: <ul style="list-style-type: none"> <li>- There are no PID/CRC5 errors in both the EXT token and the LPM token (else ERROR)</li> <li>- A valid bLinkState = 0001B (L1) is received in the LPM transaction (else STALL)</li> <li>- No data is pending in the Transmit queue (else NYET)</li> </ul> </li> <li>■ 0: NYET The pre-programmed software bit is overridden for response to LPM token when: <ul style="list-style-type: none"> <li>- The received bLinkState is not L1 (STALL response)</li> <li>- An error is detected in either of the LPM token packets due to corruption (ERROR response).</li> </ul> </li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NYET_RESP): The core responds with a NYET when an error is detected in either of the LPM token packets due to corruption</li> <li>■ 0x1 (ACK_RESP): The core responds with an ACK only on a successful LPM transaction</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM==1</p>

**Table 7-27 Fields for Register: GLPMCFG (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
0	LPMCap	R/W	<p>LPM-Capable (LPMCap)  The application uses this bit to control the controller 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. If GLPMCFG.LPMCap is 1'b0, the software must not set any of the remaining fields in the GLPMCFG register and these fields should hold their Reset values.</p> <ul style="list-style-type: none"> <li>■ 1'b0: LPM capability is not enabled.</li> <li>■ 1'b1: LPM capability is enabled.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): LPM capability is not enabled</li> <li>■ 0x1 (ENABLED): LPM capability is enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM==1</p>

7.1.23 GPWRDN

- **Name:** Global Power Down Register
  - **Description:** This is the external Hibernation control register. This register is active only during hibernation and ADP. The application can get the status of the wakeup\_logic and control it through this register.
  - **Size:** 32 bits
  - **Offset:** 0x58
  - **Exists:** OTG\_EN\_PWRLOPT==2 || OTG\_EN\_PWRLOPT==3 || OTG\_ADPSUPPORT==1

Rsvd	31:29
MultiValidBC	28:24
ADPInt	23
BSessVld	22
IDDIG	21
LineState	20:19
StsChngIntMsk	18
StsChngInt	17
SRPDetectMsk	16
SRPDetect	15
ConnDetMsk	14
ConnectDet	13
DisconnectDetectMsk	12
DisconnectDetect	11
ResetDetMsk	10
ResetDetected	9
LineStageChangeMsk	8
LnSitsChng	7
DisableVBUS	6
PwrDnSwitch	5
PwrDnRst_n	4
PwrDnClmp	3
Restore	2
PMUActv	1
PMUIntSel	0

**Table 7-28 Fields for Register: GPWRDN**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
31:29			<b>Reserved Field: Yes</b>

**Table 7-28 Fields for Register: GPWRDN (Continued)**

Bits	Name	Memory Access	Description
28:24	MultValldBC	R	<p>MultValldBC Battery Charger ACA inputs in the following order:</p> <ul style="list-style-type: none"> <li>■ Bit 26 - rid_float</li> <li>■ Bit 25 - rid_gnd</li> <li>■ Bit 24 - rid_a</li> <li>■ Bit 23 - rid_b</li> <li>■ Bit 22 - rid_c</li> </ul> <p>These bits are present only if BC_SUPPORT = 1. Otherwise, these bits are reserved and will read 5'h0.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (RID_0): OTG device as B-device</li> <li>■ 0x1 (RID_C): OTG device as B-device, can connect</li> <li>■ 0x2 (RID_B): OTG device as B-device, cannot connect</li> <li>■ 0x4 (RID_A): OTG device as A-device</li> <li>■ 0x8 (RID_GND): ID_OTG pin is grounded</li> <li>■ 0x10 (RID_FLOAT): ID pull down when ID_OTG is floating</li> <li>■ 0x11 (RID_C_RID_FLOAT): OTG device as B-device, can connect, RID_C=1 and RID_FLOAT=1</li> <li>■ 0x12 (RID_B_RID_FLOAT): OTG device as B-device, cannot connect, RID_B=1 and RID_FLOAT=1</li> <li>■ 0xc (RID_A_RID_GND): OTG device as A-device, RID_A=1 and RID_GND=1</li> <li>■ 0x1f (RID_1): OTG device as A-device</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_BC_SUPPORT==1</p>
23	ADPInt	R/W1C	<p>ADP Interrupt (ADPInt) This bit is set whenever there is a ADP event.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No ADP Event Interrupt</li> <li>■ 0x1 (ENABLED): ADP Event Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ADP_SUPPORT==1</p>

**Table 7-28 Fields for Register: GPWRDN (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
22	BSessVld	R	<p>B Session Valid (BSessVld)  This field reflects the B session valid status signal from the PHY.</p> <ul style="list-style-type: none"> <li>■ 1'b0: B-Valid is 0.</li> <li>■ 1'b1: B-Valid is 1.</li> </ul> <p>This bit is valid only when GPWRDN.PMUActv is 1.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTVALID): B_Valid is 0</li> <li>■ 0x1 (VALID): B_Valid is 1</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE!=5    OTG_MODE!=6</p> <p><b>Testable:</b> writeAsRead</p>
21	IDDIG	R	<p>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.</p> <p>Indicates the current mode.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Host mode</li> <li>■ 1'b1: Device mode</li> </ul> <p>This bit is valid only when GPWRDN.PMUActv is 1.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Host Mode</li> <li>■ 0x1 (ENABLED): Device Mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> writeAsRead</p>

**Table 7-28 Fields for Register: GPWRDN (Continued)**

Bits	Name	Memory Access	Description
20:19	LineState	R	<p>LineState This field indicates the current linestate on USB as seen by the PMU module.</p> <ul style="list-style-type: none"> <li>■ 2'b00: DM = 0, DP = 0.</li> <li>■ 2'b01: DM = 0, DP = 1.</li> <li>■ 2'b10: DM = 1, DP = 0.</li> <li>■ 2'b11: Not-defined.</li> </ul> <p>This bit is valid only when GPWRDN.PMUActv is 1.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (LS1): Linestate on USB: DM = 0, DP = 0</li> <li>■ 0x1 (LS2): Linestate on USB: DM = 0, DP = 1</li> <li>■ 0x2 (LS3): Linestate on USB: DM = 1, DP = 0</li> <li>■ 0x3 (LS4): Linestate on USB: Not-defined</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> writeAsRead</p>
18	StsChngIntMsk	R/W	<p>StsChngIntMsk Mask for StsChng Interrupt</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOMASK): No Status Change Interrupt Mask</li> <li>■ 0x1 (MASK): Mask for Status Change Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-28 Fields for Register: GPWRDN (Continued)**

Bits	Name	Memory Access	Description
17	StsChngInt	R/W1C	<p>Status Change Interrupt (StsChngInt)  This field indicates a status change in either the IDDIG or BSessVld signal.</p> <ul style="list-style-type: none"> <li>■ 1'b0: No Status change</li> <li>■ 1'b1: Status change detected</li> </ul> <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> <p>Note: When Battery Charger is enabled and the ULPI interface is used, if StsChngInt is received and the application reads GPWRDN register and determines that it is because of a change in the value of IDDIG, then StsChngInt may be generated once again within the next few clock cycles.</p> <p>This occurs because of an ambiguity in the implementation of Battery Charger Support over the ULPI interface. After receiving the StsChngInt for the second time the application can once again read the GPWRDN register. However, this time the valueIDDIG (or BSesVld) will not have changed. The application then processes the second interrupt but no further action will be required as a result.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No Status change</li> <li>■ 0x1 (ENABLED): Status change detected</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
16	SRPDetectMsk	R/W	<p>SRPDetectMsk  Mask for SRPDetect Interrupt</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOMASK): No SRPDetect Interrupt Mask</li> <li>■ 0x1 (MASK): Mask for SRPDetect Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE!= 3    OTG_MODE!=4</p>

**Table 7-28 Fields for Register: GPWRDN (Continued)**

Bits	Name	Memory Access	Description
15	SRPDetect	R/W1C	<p>SRPDetect</p> <p>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> <ul style="list-style-type: none"> <li>■ 1'b0: SRP not detected</li> <li>■ 1'b1: SRP detected</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): SRP not detected</li> <li>■ 0x1 (ENABLED): SRP detected</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE!=3    OTG_MODE!=4</p>
14	ConnDetMsk	R/W	<p>ConnDetMsk</p> <p>Mask for ConnectDet interrupt</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOMASK): No ConnectDet Interrupt Mask</li> <li>■ 0x1 (MASK): Mask for ConnectDet Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_PWR0PT==2</p>
13	ConnectDet	R/W1C	<p>ConnectDet</p> <p>This field indicates that a new connect has been detected</p> <ul style="list-style-type: none"> <li>■ 1'b0: Connect not detected</li> <li>■ 1'b1: Connect detected</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Connect not detected</li> <li>■ 0x1 (ENABLED): Connect detected</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_PWR0PT==2</p>
12	DisconnectDetectMsk	R/W	<p>DisconnectDetectMsk</p> <p>Mask For DisconnectDetect Interrupt</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOMASK): No DisconnectDetect Interrupt Mask</li> <li>■ 0x1 (MASK): Mask for DisconnectDetect Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE!=3    OTG_MODE!=4) &amp;&amp; OTG_EN_PWR0PT==2</p>

**Table 7-28 Fields for Register: GPWRDN (Continued)**

Bits	Name	Memory Access	Description
11	DisconnectDetect	R/W1C	<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> <ul style="list-style-type: none"> <li>■ 1'b0: Disconnect not detected</li> <li>■ 1'b1: Disconnect detected</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disconnect not detected</li> <li>■ 0x1 (ENABLED): Disconnect detected</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE!=3    OTG_MODE!=4) &amp;&amp; OTG_EN_PWR0PT==2</p>
10	ResetDetMsk	R/W	<p>ResetDetMsk</p> <p>Mask for ResetDetected interrupt</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOMASK): No ResetDetect Interrupt Mask</li> <li>■ 0x1 (MASK): Mask for ResetDetect Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE!=5    OTG_MODE!=6) &amp;&amp; OTG_EN_PWR0PT==2</p>
9	ResetDetected	R/W1C	<p>ResetDetected</p> <p>This field indicates that Reset has been detected by the PMU module. This field generates an interrupt.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Reset Not Detected</li> <li>■ 1'b1: Reset Detected</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Reset not detected</li> <li>■ 0x1 (ENABLED): Reset detected</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE!=5    OTG_MODE!=6) &amp;&amp; OTG_EN_PWR0PT==2</p>

**Table 7-28 Fields for Register: GPWRDN (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
8	LineStageChangeMsk	R/W	<p>LineStageChangeMsk Mask for LineStateChange interrupt.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOMASK): No LineStateChange Interrupt Mask</li> <li>■ 0x1 (MASK): Mask for LineStateChange Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_PWR OPT==2</p>
7	LnStsChng	R/W1C	<p>Line State Change (LnStsChng) 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> <ul style="list-style-type: none"> <li>■ 1'b0: No LineState change on USB</li> <li>■ 1'b1: LineState change on USB</li> </ul> <p>This bit is valid only when GPWRDN.PMUActv is 1.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No LineState change on USB</li> <li>■ 0x1 (ENABLED): LineState change on USB</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_PWR OPT==2</p>

**Table 7-28 Fields for Register: GPWRDN (Continued)**

Bits	Name	Memory Access	Description
6	DisableVBUS	R/W	<p>DisableVBUS</p> <p><b>Host Mode:</b></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> <ul style="list-style-type: none"> <li>■ 1'b0: HPRT0.PrtPwr was not programmed to 0.</li> <li>■ 1'b1: HPRT0.PrtPwr was programmed to 0.</li> </ul> <p><b>Device Mode:</b></p> <p>The application must program this bit to inform the PMU whether the bvalid valid signal is high (session valid) or low (session end) whenever the core is switched off.</p> <ul style="list-style-type: none"> <li>■ 1'b0: bvalid signal is High (Session Valid)</li> <li>■ 1'b1: bvalid signal is Low (Session End)</li> </ul> <p>This bit is valid only when GPWRDN.PMUActv is 1.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Host mode:HPRT0.PrtPwr was not programmed to 0, and in Device mode:Session Valid</li> <li>■ 0x1 (ENABLED): Host mode:HPRT0.PrtPwr was programmed to 0 and in Device mode:Session End</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_MODE!= 3    OTG_MODE!=4</p>
5	PwrDnSwtch	R/W	<p>Power Down Switch (PwrDnSwtch)</p> <p>This bit indicates to the controller whether the VDD switch is in ON/OFF state.</p> <ul style="list-style-type: none"> <li>■ 1'b0: The controller is in ON state</li> <li>■ 1'b1: The controller is in OFF state</li> </ul> <p>Note: This bit must not be written to during normal mode of operation.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ON): The controller is in ON state</li> <li>■ 0x1 (OFF): The controller is in OFF state</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-28 Fields for Register: GPWRDN (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
4	PwrDnRst_n	R/W	<p>Power Down ResetN (PwrDnRst_n)  The application must program this bit to reset the core during the Hibernation exit process or during ADP when powering up the core (in case the core was powered off during ADP process).</p> <ul style="list-style-type: none"> <li>■ 1'b1: The controller is in normal operation</li> <li>■ 1'b0: reset the controller</li> </ul> <p>Note: This bit must not be written to during normal mode of operation.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Reset the controller</li> <li>■ 0x1 (ENABLE): The controller is in normal operation</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p>
3	PwrDnClmp	R/W	<p>Power Down Clamp (PwrDnClmp)  The application must program this bit to enable or disable the clamps to all the outputs of the core module to prevent the corruption of other active logic.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Disable PMU power clamp</li> <li>■ 1'b1: Enable PMU power clamp</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Disable PMU power clamp</li> <li>■ 0x1 (ENABLE): Enable PMU power clamp</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
2	Restore	R/W	<p>Restore  The application should program this bit to enable or disable restore mode from the PMU module.</p> <ul style="list-style-type: none"> <li>■ 1'b0: The controller in normal mode of operation</li> <li>■ 1'b1: The controller in restore mode</li> </ul> <p>Note: This bit must not be written to during normal mode of operation.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): The controller in normal mode of operation</li> <li>■ 0x1 (ENABLE): The controller in Restore mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_PWROPT==2</p>

**Table 7-28 Fields for Register: GPWRDN (Continued)**

Bits	Name	Memory Access	Description
1	PMUActv	R/W	<p>PMU Active (PMUActv)  This is bit is to enable or disable the PMU logic.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Disable PMU module</li> <li>■ 1'b1: Enable PMU module</li> </ul> <p>Note: This bit must not be written to during normal mode of operation.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Disable PMU module</li> <li>■ 0x1 (ENABLE): Enable PMU module</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
0	PMUIntSel	R/W	<p>PMU Interrupt Select (PMUIntSel)  A write to this bit with 1'b1 enables the PMU to generate interrupts to the application. During this state all interrupts from the DWC_otg_core module are blocked to the application.</p> <p>Note: This bit must be set to 1'b1 before the core is put into hibernation.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Internal DWC_otg_core interrupt is selected</li> <li>■ 1'b1: External DWC_otg_pmu interrupt is selected</li> </ul> <p>Note: This bit must not be written to during normal mode of operation.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Internal DWC_otg_core interrupt is selected</li> <li>■ 0x1 (ENABLE): External DWC_otg_pmu interrupt is selected</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.24 GDFIFO CFG

- **Name:** Global DFIFO Configuration Register
- **Description:** Register to configure the DFIFOs for the controller.
- **Size:** 32 bits
- **Offset:** 0x5c
- **Exists:** OTG\_ENDED\_TX\_FIFO==1



**Table 7-29 Fields for Register: GDFIFO CFG**

Bits	Name	Memory Access	Description
31:16	EPInfoBaseAddr	R/W	<p>EPInfoBaseAddr This field provides the start address of the EP info controller. <b>Value After Reset:</b> 0x400 <b>Exists:</b> Always</p>
15:0	GDFIFO Cfg	R/W	<p>GDFIFO Cfg This field is for dynamic programming of the DFIFO Size. This value takes effect only when the application programs a non zero value to this register. The value programmed must conform to the guidelines described in 'FIFO RAM Allocation'. The core does not have any corrective logic if the FIFO sizes are programmed incorrectly. <b>Value After Reset:</b> 0x400 <b>Exists:</b> Always</p>

## 7.1.25 GADPCTL

- **Name:** ADP Timer, Control and Status Register
  - **Description:** This register is maintained in the PMU module. These register values are used for deciding the timing values by the ADP controller. This register is available only if the ADP controller logic is present within the HS OTG Controller. If the ADP logic is external to the HS OTG Controller, this register is reserved and the register contents are zero. For more information about ADP controller options, see "ADP Programming Flow when ADP Controller Logic is Supplied with the Core " in the Programming Guide.
  - **Size:** 32 bits
  - **Offset:** 0x60
  - **Exists:** OTG ADP SUPPORT==1

Rsvd	31:29
AR	28:27
AdpToutMsk	26
AdpSnsIntMsk	25
AdpPrbIntMsk	24
AdpToutInt	23
AdpSnsInt	22
AdpPrbInt	21
ADPEN	20
ADPRES	19
ENAsns	18
EnaPrb	17
RTIM	16:6
PrbPer	5:4
PrbDelta	3:2
PrbDschg	1:0

**Table 7-30 Fields for Register: GADPCTL**

Bits	Name	Memory Access	Description
31:29			<b>Reserved Field:</b> Yes
28:27	AR	R/W	<p>Access Request (AR)</p> <ul style="list-style-type: none"> <li>■ 2'b00 Read/Write Valid (updated by the core)</li> <li>■ 2'b01 Read</li> <li>■ 2'b10 Write</li> <li>■ 2'b11 - Reserved</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (Request1): Read/Write</li> <li>■ 0x1 (Request2): Read</li> <li>■ 0x2 (Request3): Write</li> <li>■ 0x3 (Request4): Reserved</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-30 Fields for Register: GADPCTL (Continued)**

Bits	Name	Memory Access	Description
26	AdpToutMsk	R/W	<p>ADP Timeout Interrupt Mask (AdpToutMsk) When this bit is set, it unmasks the interrupt because of AdpTmouInt. This bit is valid only if OTG_Ver = 1'b1(GOTGCTL[20]).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask ADP Timeout Interrupt</li> <li>■ 0x1 (NOMASK): Unmask ADP Timeout Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
25	AdpSnsIntMsk	R/W	<p>ADP Sense Interrupt Mask (AdpSnsIntMsk) When this bit is set, it unmasks the interrupt due to AdpSnsInt. This bit is valid only if OTG_Ver = 1'b1(GOTGCTL[20]).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask ADP Sense Interrupt</li> <li>■ 0x1 (NOMASK): Unmask ADP Sense Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
24	AdpPrbIntMsk	R/W	<p>ADP Probe Interrupt (AdpPrbInt) 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 OTGVer = 1'b1.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask ADP Probe Interrupt</li> <li>■ 0x1 (NOMASK): Unmask ADP Probe Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
23	AdpToutInt	R/W1C	<p>ADP Timeout Interrupt (AdpToutInt) 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 OTGVer = 1'b1.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): ADP Timeout Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-30 Fields for Register: GADPCTL (Continued)**

Bits	Name	Memory Access	Description
22	AdpSnsInt	R/W1C	<p>ADP Sense Interrupt (AdpSnsInt) 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 OTGVer = 1'b1 (GOTGCTL[20]).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): ADP Sense Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
21	AdpPrbInt	R/W1C	<p>ADP Probe Interrupt (AdpPrbInt) When this bit is set, it means that the VBUS voltage is greater than VADP_PRB or VadpPrb is reached. This bit is valid only if OTGVer = 1'b1 (GOTGCTL[20]).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): ADP Probe Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
20	ADPEn	R/W	<p>ADP Enable (ADPEn) 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]).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Not active</li> <li>■ 0x1 (ENABLE): Core does ADP Probing or Sensing</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
19	ADPRes	R/W1S	<p>ADP Reset (ADPRes) 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]).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Not active</li> <li>■ 0x1 (ENABLE): ADP controller is reset</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-30 Fields for Register: GADPCTL (Continued)**

Bits	Name	Memory Access	Description
18	EnaSns	R/W	<p>Enable Sense (EnaSns) When programmed to 1'b1, the core performs a sense operation. This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Not active</li> <li>■ 0x1 (ENABLE): Core performs sense operation</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
17	EnaPrb	R/W	<p>Enable Probe (EnaPrb) When programmed to 1'b1, the core performs a probe operation. This bit is valid only if OTG_Ver = 1'b1 (GOTGCTL[20]).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Not active</li> <li>■ 0x1 (ENABLE): Core performs probe operation</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
16:6	RTIM	R	<p>RAMP TIME (RTIM) 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"> <li>■ 0x000 - 1 cycles</li> <li>■ 0x001 - 2 cycles</li> <li>■ 0x002 - 3 cycles, and so on till</li> <li>■ 0x7FF - 2048 cycles</li> </ul> <p>A time of 1024 cycles at 32 kHz corresponds to a time of 32 msec. (Note for scaledown ramp_timeout =</p> <ul style="list-style-type: none"> <li>■ prb_delta == 2'b00 =&gt; 200 cycles.</li> <li>■ prb_delta == 2'b01 =&gt; 100 cycles.</li> <li>■ prb_delta == 2'b01 =&gt; 50 cycles.</li> <li>■ prb_delta == 2'b01 =&gt; 25 cycles.)</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (RTIM1): 1 cycle</li> <li>■ 0x1 (RTIM2): 2 cycles</li> <li>■ 0x2 (RTIM3): 3 cycles</li> <li>■ 0x7ff (RTIM4): 2048 cycles</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-30 Fields for Register: GADPCTL (Continued)**

Bits	Name	Memory Access	Description
5:4	PrbPer	R/W	<p>Probe Period (PrbPer)  These bits sets the TadpPrd as follows:</p> <ul style="list-style-type: none"> <li>■ 2'b00 - 0.625 to 0.925 sec (typical 0.775 sec)</li> <li>■ 2'b01 - 1.25 to 1.85 sec (typical 1.55 sec)</li> <li>■ 2'b10 - 1.9 to 2.6 sec (typical 2.275 sec)</li> <li>■ 2'b11 - Reserved  (PrbPer is also scaledown <ul style="list-style-type: none"> <li>■ prb_per== 2'b00 =&gt; 400 ADP clocks</li> <li>■ prb_per== 2'b01 =&gt; 600 ADP clocks</li> <li>■ prb_per== 2'b10 =&gt; 800 ADP clocks</li> <li>■ prb_per==2'b11 =&gt; 1000 ADP clocks)</li> </ul> </li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (TADP_PRB_0.6_0.9SEC): TadpPrd: 0.625 to 0.925 sec</li> <li>■ 0x1 (TADP_PRB_1.2_1.8SEC): TadpPrd: 1.25 to 1.85 sec</li> <li>■ 0x2 (TADP_PRB_1.9_2.6SEC): TadpPrd: 1.9 to 2.6 sec</li> <li>■ 0x3 (RESERVED): Reserved</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
3:2	PrbDelta	R/W	<p>Probe Delta (PrbDelta)  These bits set the resolution for RTIM value. The bits are defined in units of 32 kHz clock cycles as follows:</p> <ul style="list-style-type: none"> <li>■ 2'b00: 1 cycles</li> <li>■ 2'b01: 2 cycles</li> <li>■ 2'b10: 3 cycles</li> <li>■ 2'b11: 4 cycles</li> </ul> <p>For example, if this value is chosen to 2'b01, it means that RTIM increments for every three 32Khz clock cycles.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (VALUE1): 1 cycle</li> <li>■ 0x1 (VALUE2): 2 cycles</li> <li>■ 0x2 (VALUE3): 3 cycles</li> <li>■ 0x3 (VALUE4): 4 cycles</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-30 Fields for Register: GADPCTL (Continued)**

Bits	Name	Memory Access	Description
1:0	PrbDschg	R/W	<p>Probe Discharge (PrbDschg)  These bits set the times for TADP_DSCHG. These bits are defined as follows:</p> <ul style="list-style-type: none"> <li>■ 2'b00: 4 msec (Scaledown 2 32Khz clock cycles)</li> <li>■ 2'b01: 8 msec (Scaledown 4 32Khz clock cycles)</li> <li>■ 2'b10: 16 msec (Scaledown 8 32Khz clock cycles)</li> <li>■ 2'b11: 32 msec (Scaledown 16 32Khz clock cycles)</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (VALUE1): Probe Discharge time: 4 msec</li> <li>■ 0x1 (VALUE2): Probe Discharge time: 8 msec</li> <li>■ 0x2 (VALUE3): Probe Discharge time: 16 msec</li> <li>■ 0x3 (VALUE4): Probe Discharge time: 32 msec</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

### 7.1.26 GREFCLK

- **Name:** ref\_clk Control Register
- **Description:** Register to control the ref\_clk properties for the controller.
- **Size:** 32 bits
- **Offset:** 0x64
- **Exists:** OTG\_ADC30\_REF\_CLK\_EN==1

REFCLKPER	31:15			
RefclkMode	14			
RESERVED	13:10			
SOF_CNT_WKUP_ALERT	9:0			

Table 7-31 Fields for Register: GREFCLK

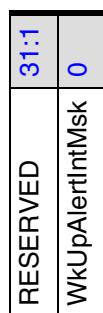
Bits	Name	Memory Access	Description
31:15	REFCLKPER	R/W	<p>This bit indicates the period of ref_clk in terms of pico seconds.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>■ The default value of this field is 'd0.</li> <li>■ The period of ref_clk should be an integer multiple of 125us.</li> <li>■ The minimum frequency supported is 12MHz.</li> </ul> <p>Other supported frequencies are 16, 17, 19.2, 20, 24, 30, and 40MHz.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-31 Fields for Register: GREFCLK (Continued)**

Bits	Name	Memory Access	Description
14	RefclkMode	R/W	<p>This bit is used to enable or disable ref_clk mode of operation.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>■ The default value of this field is 'd0.</li> <li>■ When this field is disabled, DCTL.ServInt cannot be set to 1.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (ENABLE): Controller uses ref_clk to run internal micro-frame timers</li> <li>■ 0x0 (DISABLE): Controller uses phy_clk to run internal micro-frame timers</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
13:10	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
9:0	SOF_CNT_WKUP_ALERT	R/W	<p>This bit indicates the number of SOF's after which the controller should generate an interrupt if the device had been in L1 state until that period.</p> <p>The interrupt is used by software to initiate remote wakeup in the controller in order to sync to the uF number in the host. For more details, refer to the "Auto Remote Wakeup Feature" section of the Programming Guide.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.27 GINTMSK2

- **Name:** Interrupt Mask Register 2
  - **Description:** This register works with the Interrupt Register (GINTSTS2) to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the GINTSTS2 register bit corresponding to that interrupt is still set.
- Note:** The fields of this register change depending on host or device mode.
- **Size:** 32 bits
  - **Offset:** 0x68
  - **Exists:** Always



**Table 7-32 Fields for Register: GINTMSK2**

Bits	Name	Memory Access	Description
31:1	RESERVED	R	<b>RESERVED</b> <b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead
0	WkUpAlertIntMsk	R/W	Mode: Device Remote WakeUp Alert Interrupt Mask This interrupt is used to alert the application to initiate Remote WakeUp sequence. <b>Values:</b> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Remote WakeUp Alert Interrupt</li> <li>■ 0x1 (NOMASK): Unmask Remote WakeUp Alert Interrupt</li> </ul> <b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_ADC30_REF_CLK_EN==1

### 7.1.28 GINTSTS2

- **Name:** Interrupt Register 2
- **Description:** This register interrupts the application for system-level events in the current mode (Device mode or Host mode).

Some of the bits in this register are valid only in Host mode, while others are valid in Device mode only. This register also indicates the current mode. To clear the interrupt status bits of type R\_SS\_WC, the application must write 1'b1 to the bit.

The application must clear the GINTSTS2 register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

- **Size:** 32 bits
- **Offset:** 0x6c
- **Exists:** Always



**Table 7-33 Fields for Register: GINTSTS2**

Bits	Name	Memory Access	Description
31:1	RESERVED	R	<b>RESERVED</b> <b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead
0	WkUpAlertInt	R/W1C	Mode: Device Remote WakeUp Alert Interrupt Mask This interrupt is used to alert the application to initiate Remote WakeUp sequence. <b>Values:</b> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not Active</li> <li>■ 0x1 (ACTIVE): Remote WakeUp Alert Interrupt detected</li> </ul> <b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_ADC30_REF_CLK_EN==1

### 7.1.29 HPTXFSIZ

- **Name:** Host Periodic Transmit FIFO Size Register
- **Description:** This register holds the size and the memory start address of the Periodic TxFIFO.
- Note:** Read the reset value of this register only after the following conditions:
  - If IDDIG\_FILTER is disabled, read only after PHY clock is stable.
  - If IDDIG\_FILTER is enabled, read only after the filter timer expires.
- **Size:** 32 bits
- **Offset:** 0x100
- **Exists:** (OTG\_MODE != 3 && OTG\_MODE != 4 && OTG\_DFIFO\_DYNAMIC != 0)

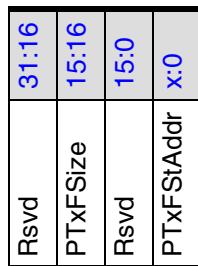


Table 7-34 Fields for Register: HPTXFSIZ

Bits	Name	Memory Access	Description
31:16			<b>Reserved Field:</b> Yes
15:16	PTxFSize	OTG_DFI FO_DYN AMIC == 1 ? R/W : R	<p>Host Periodic TxFIFO Depth (PTxFSize) This value is in terms of 32-bit words.</p> <ul style="list-style-type: none"> <li>■ Minimum value is 16</li> <li>■ Maximum value is 32,768</li> </ul> <p>The power-on reset value of this register is specified as the Largest Host Mode Periodic Tx Data FIFO Depth.</p> <ul style="list-style-type: none"> <li>■ If Enable Dynamic FIFO Sizing? was deselected in coreConsultant (parameter OTG_DFIFO_DYNAMIC = 0), these flops are optimized, and reads return the power-on value.</li> <li>■ If Enable Dynamic FIFO Sizing? was selected in coreConsultant (parameter OTG_DFIFO_DYNAMIC = 1), you can write a new value in this field.</li> </ul> <p>Programmed values must not exceed the power-on value set in coreConsultant.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-34 Fields for Register: HPTXFSIZ (Continued)**

Bits	Name	Memory Access	Description
15:0			<b>Reserved Field:</b> Yes
x:0	PTxFStAddr	OTG_DFI FO_DYN AMIC == 1 ? R/W : R	<p>Host Periodic TxFIFO Start Address (PTxFStAddr)  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.These parameters are:</p> <p>In shared FIFO operation:</p> <ul style="list-style-type: none"> <li>■ OTG_RX_DFIFO_DEPTH + OTG_TX_NPERIO_DFIFO_DEPTH</li> </ul> <p>In dedicated FIFO mode:</p> <ul style="list-style-type: none"> <li>■ OTG_RX_DFIFO_DEPTH + OTG_TX_HNPERIO_DFIFO_DEPTH If Enable Dynamic FIFO Sizing? was deselected in coreConsultant (parameter OTG_DFIFO_DYNAMIC = 0), these flops are optimized, and reads return the power-on value. If Enable Dynamic FIFO Sizing? was selected in coreConsultant (parameter OTG_DFIFO_DYNAMIC = 1), you can write a new value in this field.</li> </ul> <p>Programmed values must not exceed the power-on value set in coreConsultant.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Range Variable[x]:</b> -1</p>

### 7.1.30 DPTXFSIZi (for i = 1; i <= OTG\_NUM\_PERIO\_EPS-1)

- **Name:** Device Periodic Transmit FIFO-i Size Register
- **Description:** This register is valid only in shared FIFO operation (OTG\_ENDED\_TX\_FIFO = 0).

This register holds the memory start address of each periodic TxFIFO to be implemented in Device mode, as shown in "PMU Logic" and "PMU Sync Module" sections in the databook. Each periodic FIFO holds the data for one periodic IN endpoint. This register is repeated for each periodic FIFO instantiated.

- **Size:** 32 bits
- **Offset:** 0x104 + (i-1)\*004
- **Exists:** ((OTG\_MODE!=5) && (OTG\_MODE!=6)) && (OTG\_NUM\_PERIO\_EPS>=i) && (OTG\_ENDED\_TX\_FIFO==0)



**Table 7-35 Fields for Register: DPTXFSIZi (for i = 1; i <= OTG\_NUM\_PERIO\_EPS-1)**

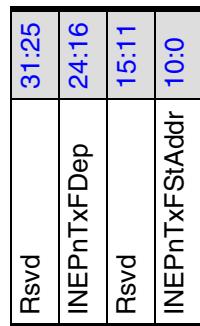
Bits	Name	Memory Access	Description
31:16	DPTxFSIZE	R	<p>Device Periodic TxFIFO Size (DPTxFSIZE) This value is in terms of 32-bit words.</p> <ul style="list-style-type: none"> <li>■ Minimum value is 4</li> <li>■ Maximum value is 768</li> </ul> <p>The value of this register is the Largest Device Mode Periodic Tx Data FIFO Depth</p> <p><b>Value After Reset:</b> 0x300</p> <p><b>Exists:</b> Always</p>
15:12			<b>Reserved Field:</b> Yes

**Table 7-35 Fields for Register: DPTXFSI<sub>i</sub> (for i = 1; i <= OTG\_NUM\_PERIO\_EPS-1) (Continued)**

Bits	Name	Memory Access	Description
11:0	DPTxFStAddr	OTG_DFI FO_DYN AMIC == 1 ? R/W : R	<p>Device Periodic TxFIFO RAM Start Address (DPTxFStAddr)  This register field holds the start address in the RAM for this periodic FIFO. The power-on reset value of this register is the sum of the Largest Rx Data FIFO Depth, Largest Non-periodic Tx Data FIFO Depth, and all lower numbered Largest Device Mode Periodic Tx Data FIFOOn Depth.specified during coreConsultant configuration.</p> <p>The formula used is: OTG_RX_DFI<sub>0</sub>_DEPTH + OTG_TX_NPERIO_DFI<sub>0</sub>_DEPTH + SUM of OTG_TX_DPERIO_DFI<sub>0</sub>_DEPTH_&gt;i (where x=1 to n-1). When n = 1, the above expression becomes OTG_RX_DFI<sub>0</sub>_DEPTH + OTG_TX_NPERIO_DFI<sub>0</sub>_DEPTH. If at POR, the calculated value (C) exceeds 65535, then the Reset value becomes Reset Value(A) = (C-65536).</p> <ul style="list-style-type: none"> <li>■ If Enable Dynamic FIFO Sizing is not selected in coreConsultant (OTG_DFI<sub>0</sub>_DYNAMIC = 0), this field is read-only and read value is the power-on reset value.</li> <li>■ If Enable Dynamic FIFO Sizing is selected in coreConsultant (OTG_DFI<sub>0</sub>_DYNAMIC = 1), and you have calculated/programmed new values for the RxFIFO Non-periodic TxFIFO, or device Periodic TxFIFOs, you can program their sum according to the above formula.</li> </ul> <p><b>Value After Reset:</b> 0x800  <b>Exists:</b> Always</p>

### 7.1.31 DIEPTXFi (for i = 1; i <= OTG\_NUM\_IN\_EPS-1)

- **Name:** Device IN Endpoint Transmit FIFO Size Register
- **Description:** This register is valid only in dedicated FIFO mode (OTG\_ENDED\_TX\_FIFO=1). It holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for instantiated IN endpoint FIFOs 1 to 15. For IN endpoint FIFO 0, use GNPTXFSIZ register for programming the size and memory start address.
- **Size:** 32 bits
- **Offset:** 0x104 + (i-1)\*004
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_IN\_EPS > 1) && (OTG\_ENDED\_TX\_FIFO==1)



**Table 7-36 Fields for Register: DIEPTXFi (for i = 1; i <= OTG\_NUM\_IN\_EPS-1)**

Bits	Name	Memory Access	Description
31:25			<b>Reserved Field:</b> Yes

**Table 7-36 Fields for Register: DIEPTXFi (for i = 1; i <= OTG\_NUM\_IN\_EPS-1) (Continued)**

Bits	Name	Memory Access	Description
24:16	INEPnTxFDep	OTG_DFI FO_DYN AMIC == 1 ? R/W : R	<p>IN Endpoint Tx FIFO Depth (INEPnTxFDep) This value is in terms of 32-bit words.</p> <ul style="list-style-type: none"> <li>■ Minimum value is 16</li> <li>■ Maximum value is 32,768</li> </ul> <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) during coreConsultant configuration (0 &lt; i &lt;= 15).</p> <ul style="list-style-type: none"> <li>■ If "Enable Dynamic FIFO Sizing?" was deselected in coreConsultant (parameter OTG_DFIFO_DYNAMIC = 0), these flops are optimized, and reads return the power-on value.</li> <li>■ If "Enable Dynamic FIFO Sizing?" was selected in coreConsultant (parameter OTG_DFIFO_DYNAMIC = 1), you can write a new value in this field.</li> </ul> <p>Programmed values must not exceed the power-on value.</p> <p><b>Value After Reset:</b> 0x100</p> <p><b>Exists:</b> Always</p>
15:11			<b>Reserved Field:</b> Yes

**Table 7-36 Fields for Register: DIEPTXFi (for i = 1; i <= OTG\_NUM\_IN\_EPS-1) (Continued)**

Bits	Name	Memory Access	Description
10:0	INEPnTxFStAddr	OTG_DFI FO_DYN AMIC == 1 ? R/W : R	<p>IN Endpoint FIFOOn Transmit RAM Start Address (INEPnTxFStAddr)</p> <p>This field contains the memory start address for IN endpoint Transmit FIFOOn (0&lt;n&lt; = 15). The power-on reset value of this register is specified as the Largest Rx Data FIFO Depth. The power-on reset value of this register is calculated according to the following formula: OTG_RX_DFIIFO_DEPTH + SUM of OTG_TX_DINEP_DFIIFO_DEPTH_‘i’ (where x = 0 to n 1) If at POR the calculated value (C) exceeds 65535, then the Reset value becomes Reset Value(A) = (C 65536). Example: If start address of IN endpoint FIFO 1 is OTG_RX_DFIIFO_DEPTH + OTG_TX_DINEP_DFIIFO_DEPTH_0 and start address of IN endpoint FIFO 2 is OTG_RX_DFIIFO_DEPTH + OTG_TX_DINEP_DFIIFO_DEPTH_0 + OTG_TX_DINEP_DFIIFO_DEPTH_1.</p> <ul style="list-style-type: none"> <li>■ If Enable Dynamic FIFO Sizing is deselected in coreConsultant (OTG_DFIIFO_DYNAMIC = 0), this field is read-only and read value is the power-on reset value.</li> <li>■ If Enable Dynamic FIFO Sizing is selected in coreConsultant (OTG_DFIIFO_DYNAMIC = 1), and you have calculated or programmed a new value for RxFIFO depth or TX FIFO depths, you can program their values according to the above formula. Programmed values must not exceed the power-on value set in coreConsultant.</li> </ul> <p><b>Value After Reset:</b> 0x500</p> <p><b>Exists:</b> Always</p>

### 7.1.32 HCFG

- **Name:** Host Configuration Register
- **Description:** This register is used to configure the controller in Host mode.
- **Size:** 32 bits
- **Offset:** 0x400
- **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==5 || OTG\_MODE == 6

ModeChTimEn	31	30:27
RESERVED		26
PerSchedEna		25:24
FrlstEn		23
DescDMA		22:17
RESERVED1		16
dis_tx_ipgap_dly_check		15:8
ResValid		7
Ena32KHzS		6:3
RESERVED2		2
FSLSSupp		1:0
FSLSPclkSel		

**Table 7-37 Fields for Register: HCFG**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
31	ModeChTimEn	R/W	<p>Mode Change Ready Timer Enable (ModeChTimEn)  This bit is used to enable/disable the Host core to wait 200 PHY clock cycles at the end of Resume to change the opmode signal to the PHY to 00 after Suspend or LPM.</p> <ul style="list-style-type: none"> <li>■ 1'b0 : The Host core waits for either 200 PHY clock cycles or a linestate of SE0 at the end of resume to change the opmode from 2'b10 to 2'b00</li> <li>■ 1'b1 : The Host core waits only for a linestate of SE0 at the end of resume to change the opmode from 2'b10 to 2'b00.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ENABLED):  The Host core waits for either 200 PHY clock cycles or a linestate of SE0 at the end of resume to change the opmode from 0x2 to 0x0</li> <li>■ 0x1 (DISABLED):  The Host core waits only for a linestate of SE0 at the end of resume to change the opmode from 0x2 to 0x0</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
30:27	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> 0  <b>Testable:</b> writeAsRead  <b>Write Constraint:</b> writeAsRead</p>

**Table 7-37 Fields for Register: HCFG (Continued)**

Bits	Name	Memory Access	Description
26	PerSchedEna	R/W	<p>Enable Periodic Scheduling (PerSchedEna): Applicable in host DDMA mode only. Enables periodic scheduling within the core. Initially, the bit is reset. The core will not process any periodic channels.</p> <p>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.</p> <p>As soon as this bit is reset, the core will get ready to stop scheduling periodic channels and resets HCFG.PerSchedStat.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disables periodic scheduling within the core</li> <li>■ 0x1 (ENABLED): Enables periodic scheduling within the core</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA == 1</p> <p><b>Testable:</b> writeAsRead</p>
25:24	FrListEn	R/W	<p>Frame List Entries(FrListEn)</p> <p>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> <ul style="list-style-type: none"> <li>■ 2'b00: 8 Entries</li> <li>■ 2'b01: 16 Entries</li> <li>■ 2'b10: 32 Entries</li> <li>■ 2'b11: 64 Entries</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ENTRY8): 8 Entries</li> <li>■ 0x1 (ENTRY16): 16 Entries</li> <li>■ 0x2 (ENTRY32): 32 Entries</li> <li>■ 0x3 (ENTRY63): 64 Entries</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA == 1</p>

**Table 7-37 Fields for Register: HCFG (Continued)**

Bits	Name	Memory Access	Description
23	DescDMA	R/W	<p>Enable Scatter/gather DMA in Host mode (DescDMA)  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.</p> <p>Note: This bit must be modified only once after a reset.  The following combinations are available for programming:</p> <ul style="list-style-type: none"> <li>■ GAHBCFG.DMAEn=0,HCFG.DescDMA=0 =&gt; Slave mode</li> <li>■ GAHBCFG.DMAEn=0,HCFG.DescDMA=1 =&gt; Invalid</li> <li>■ GAHBCFG.DMAEn=1,HCFG.DescDMA=0 =&gt; Buffered DMA mode</li> <li>■ GAHBCFG.DMAEn=1,HCFG.DescDMA=1 =&gt; Scatter/Gather DMA mode</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Scatter/Gather DMA in Host mode is disabled</li> <li>■ 0x1 (ENABLED): Scatter/Gather DMA selected</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_DESC_DMA == 1</p>
22:17	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> 0  <b>Testable:</b> writeAsRead  <b>Write Constraint:</b> writeAsRead</p>
16	dis_tx_ipgap_dly_check	R/W	<p>Disable Linestate check for Token to Token/Data interpacket gap delay calculation (dis_tx_ipgap_dly_check)</p> <p>Applicable only in HS mode of operation. When enabled, the controller implements counter based logic to detect the end of token transmission, otherwise controller monitors the linestate for end of token transmission,to start the Token to Token and Token to Data interpacket delay counter.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Enable Linestate check</li> <li>■ 0x1 (ENABLE): Disable Linestate check</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_FSPHY_INTERFACE == 0 &amp;&amp; OTG_EN_DESC_DMA == 1</p>

**Table 7-37 Fields for Register: HCFG (Continued)**

Bits	Name	Memory Access	Description
15:8	ResValid	R/W	<p>Resume Validation Period (ResValid)  This field is effective only when HCFG.Ena32KHzS is set. It will control the resume period when the core resumes from suspend. The core counts for 'ResValid' number of clock cycles to detect a valid resume when this is set.</p> <p><b>Value After Reset:</b> 0x2  <b>Exists:</b> Always</p>
7	Ena32KHzS	R/W	<p>Enable 32 KHz Suspend mode (Ena32KHzS)  This bit can be set only in FS PHY interface is selected. Else, this bit needs to be set to zero. When FS PHY interface is chosen and this bit is set, the core expects that the PHY clock during Suspend is switched from 48 MHz to 32 KHz.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): 32 KHz Suspend mode disabled</li> <li>■ 0x1 (ENABLED): 32 KHz Suspend mode enabled</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
6:3	RESERVED2	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> 0  <b>Testable:</b> writeAsRead  <b>Write Constraint:</b> writeAsRead</p>

**Table 7-37 Fields for Register: HCFG (Continued)**

Bits	Name	Memory Access	Description
2	FSLSSupp	R/W	<p>FS- and LS-Only Support (FSLSSupp)  The application uses this bit to control the core's 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> <ul style="list-style-type: none"> <li>■ 1'b0: HS/FS/LS, based on the maximum speed supported by the connected device</li> <li>■ 1'b1: FS/LS-only, even if the connected device can support HS</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (HSFSLS):  HS/FS/LS, based on the maximum speed supported by the connected device</li> <li>■ 0x1 (FSLS):  FS/LS-only, even if the connected device can support HS</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-37 Fields for Register: HCFG (Continued)**

Bits	Name	Memory Access	Description
1:0	FSLSPclkSel	R/W	<p>FS/LS PHY Clock Select (FSLSPclkSel)</p> <p>When the core is in FS Host mode</p> <ul style="list-style-type: none"> <li>■ 2'b00: PHY clock is running at 30/60 MHz</li> <li>■ 2'b01: PHY clock is running at 48 MHz</li> <li>■ Others: Reserved</li> </ul> <p>When the core is in LS Host mode</p> <ul style="list-style-type: none"> <li>■ 2'b00: PHY clock is running at 30/60 MHz. When the UTMI+/ULPI PHY Low Power mode is not selected, use 30/60 MHz.</li> <li>■ 2'b01: PHY clock is running at 48 MHz. When the UTMI+ PHY Low Power mode is selected, use 48MHz If the PHY supplies a 48 MHz clock during LS mode.</li> <li>■ 2'b10: PHY clock is running at 6 MHz. In USB 1.1 FS mode, use 6 MHz when the UTMI+ PHY Low Power mode is selected and the PHY supplies a 6 MHz clock during LS mode. If you select a 6 MHz clock during LS mode, you must do a soft reset.</li> <li>■ 2'b11: Reserved</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>■ When Core in FS mode, the internal and external clocks have the same frequency.</li> <li>■ When Core in LS mode, <ul style="list-style-type: none"> <li>- If FSLSPclkSel = 2'b00: Internal and external clocks have the same frequency</li> <li>- If FSLSPclkSel = 2'b10: Internal clock is the divided by eight version of external 48 MHz clock</li> </ul> </li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (CLK3060): PHY clock is running at 30/60 MHz</li> <li>■ 0x1 (CLK48): PHY clock is running at 48 MHz</li> <li>■ 0x2 (CLK6): PHY clock is running at 6 MHz</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.33 HFIR

- **Name:** Host Frame Interval Register
- **Description:** This register is used to control the interval between two consecutive SOFs.
- **Size:** 32 bits
- **Offset:** 0x404
- **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==5 || OTG\_MODE == 6



**Table 7-38 Fields for Register: HFIR**

Bits	Name	Memory Access	Description
31:17	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
16	HFIRRldCtrl	R/W	<p>Reload Control (HFIRRldCtrl)</p> <p>This bit allows dynamic reloading of the HFIR register during run time.</p> <ul style="list-style-type: none"> <li>■ 1'b0 : The HFIR cannot be reloaded dynamically</li> <li>■ 1'b1: the HFIR can be dynamically reloaded during runtime.</li> </ul> <p>This bit needs to be programmed during initial configuration and its value should not be changed during runtime.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): The HFIR cannot be reloaded dynamically</li> <li>■ 0x1 (ENABLED):</li> </ul> <p>The HFIR can be dynamically reloaded during runtime</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-38 Fields for Register: HFIR (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
15:0	FrInt	R/W	<p>Frame Interval (FrInt)</p> <p>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 is for 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> <ul style="list-style-type: none"> <li>■ 125 s * (PHY clock frequency for HS)</li> <li>■ 1 ms * (PHY clock frequency for FS/LS)</li> </ul> <p><b>Value After Reset:</b> 0xea60</p> <p><b>Exists:</b> Always</p>

### 7.1.34 HFNUM

- **Name:** Host Frame Number/Frame Time Remaining Register
  - **Description:** This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current (micro)frame.
- Note:** Read the reset value of this register only after the following conditions:
- If IDDIG\_FILTER is disabled, read only when the PHY clock is stable.
  - If IDDIG\_FILTER is enabled, read only after the filter timer expires.
- **Size:** 32 bits
  - **Offset:** 0x408
  - **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==5 || OTG\_MODE == 6



**Table 7-39 Fields for Register: HFNUM**

Bits	Name	Memory Access	Description
31:16	FrRem	R	<p>Frame Time Remaining (FrRem) 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.</p> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always <b>Testable:</b> untestable</p>

**Table 7-39 Fields for Register: HFNUM (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
15:0	FrNum	R	<p>Frame Number (FrNum) This field increments when a new SOF is transmitted on the USB, and is reset to 0 when it reaches 16'h3FFF.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No SOF is transmitted</li> <li>■ 0x1 (ACTIVE): SOF is transmitted</li> </ul> <p><b>Value After Reset:</b> 0x3fff</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> untestable</p>

### 7.1.35 HPTXSTS

- **Name:** Host Periodic Transmit FIFO/Queue Status Register
- **Description:** This register contains information about the Periodic Transmit Queue in the Host controller.
- **Size:** 32 bits
- **Offset:** 0x410
- **Exists:** (OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==5 || OTG\_MODE == 6) && OTG\_EN\_PERIO\_HOST ==1



**Table 7-40 Fields for Register: HPTXSTS**

Bits	Name	Memory Access	Description
31:23	PTxQTop	R	<p>Top of the Periodic Transmit Request Queue (PTxQTop)  This indicates the Entry in the Periodic Tx Request Queue that is currently being processed by the MAC.  This register is used for debugging.</p> <ul style="list-style-type: none"> <li>■ Bit [31]: Odd/Even (micro)Frame <ul style="list-style-type: none"> <li>- 1'b0: send in even (micro)Frame</li> <li>- 1'b1: send in odd (micro)Frame</li> </ul> </li> <li>■ Bits [30:27]: Channel/endpoint number</li> <li>■ Bits [26:25]: Type <ul style="list-style-type: none"> <li>- 2'b00: IN/OUT</li> <li>- 2'b01: Zero-length packet</li> <li>- 2'b10: CSPLIT</li> <li>- 2'b11: Disable channel command</li> </ul> </li> <li>■ Bit [24]: Last Periodic Entry for the selected periodic channel/endpoint</li> <li>■ Bit [23]: Terminate (last Entry for the selected channel/endpoint)</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-40 Fields for Register: HPTXSTS (Continued)**

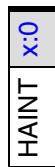
Bits	Name	Memory Access	Description
22:16	PTxQSpAvail	R	<p>Periodic Transmit Request Queue Space Available (PTxQSpAvail)      Indicates the number of free locations available to be written in the Periodic Transmit Request Queue. This queue holds both IN and OUT requests.</p> <ul style="list-style-type: none"> <li>■ 7'h0: Periodic Transmit Request Queue is full</li> <li>■ 7'h1: 1 location available</li> <li>■ 7'h2: 2 locations available</li> <li>■ n: n locations available (<math>0 \leq n \leq 16</math>)</li> <li>■ Others: Reserved</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (FULL): Periodic Transmit Request Queue is full</li> <li>■ 0x1 (FREE1): 1 location available</li> <li>■ 0x2 (FREE2): 2 locations available</li> <li>■ 0x3 (FREE3): 3 locations available</li> <li>■ 0x4 (FREE4): 4 locations available</li> <li>■ 0x5 (FREE5): 5 locations available</li> <li>■ 0x6 (FREE6): 6 locations available</li> <li>■ 0x7 (FREE7): 7 locations available</li> <li>■ 0x8 (FREE8): 8 locations available</li> <li>■ 0x9 (FREE9): 9 locations available</li> <li>■ 0xa (FREE10): 10 locations available</li> <li>■ 0xb (FREE11): 11 locations available</li> <li>■ 0xc (FREE12): 12 locations available</li> <li>■ 0xd (FREE13): 13 locations available</li> <li>■ 0xe (FREE14): 14 locations available</li> <li>■ 0xf (FREE15): 15 locations available</li> </ul> <p><b>Value After Reset:</b> 0x8  <b>Exists:</b> Always</p>

**Table 7-40 Fields for Register: HPTXSTS (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
15:0	PTxFSpAvail	R	<p>Periodic Transmit Data FIFO Space Available (PTxFSpAvail)      Indicates the number of free locations available to be written to in the Periodic TxFIFO.      Values are in terms of 32-bit words</p> <ul style="list-style-type: none"> <li>■ 16'h0 : Periodic TxFIFO is full</li> <li>■ 16'h1 : 1 word available</li> <li>■ 16'h2 : 2 words available</li> <li>■ 16'hn : n words available (where 0 &lt; n &lt; 32,768)</li> <li>■ 16'h8000 : 32,768 words</li> <li>■ Others : Reserved</li> </ul> <p><b>Value After Reset:</b> 0x400  <b>Exists:</b> Always</p>

### 7.1.36 HAINT

- **Name:** Host All Channels Interrupt Register
- **Description:** When a significant event occurs on a channel, the Host All Channels Interrupt register interrupts the application using the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt). This is shown in the "Interrupt Hierarchy" figure in the databook. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Host Channel-n Interrupt register.
- **Size:** 32 bits
- **Offset:** 0x414
- **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==5 || OTG\_MODE == 6



**Table 7-41 Fields for Register: HAINT**

Bits	Name	Memory Access	Description
x:0	HAINT	R	<p>Channel Interrupt for channel no.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Not active</li> <li>■ 0x1 (ACTIVE): Host Channel Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Range Variable[x]:</b> OTG_NUM_HOST_CHAN - 1</p>

### 7.1.37 HAINTMSK

- **Name:** Host All Channels Interrupt Mask Register
- **Description:** The Host All Channel Interrupt Mask register works with the Host All Channel Interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.
- **Size:** 32 bits
- **Offset:** 0x418
- **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==5 || OTG\_MODE == 6



**Table 7-42 Fields for Register: HAINTMSK**

Bits	Name	Memory Access	Description
x:0	HAINTMsK	R/W	<p>Channel Interrupt Mask (HAINTMsK) One bit per channel: Bit 0 for channel 0, bit 15 for channel 15</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (UNMASK): Unmask Channel interrupt</li> <li>■ 0x1 (MASK): Mask Channel interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Range Variable[x]:</b> OTG_NUM_HOST_CHAN - 1</p>

### 7.1.38 HFLBAddr

- **Name:** Host Frame List Base Address Register
- **Description:** This register is present only in case of Scatter/Gather DMA. It is implemented as flops. This register holds the starting address of the Frame list information.
- **Size:** 32 bits
- **Offset:** 0x41c
- **Exists:** OTG\_MODE != 3 && OTG\_MODE != 4 && OTG\_EN\_DESC\_DMA == 1



**Table 7-43 Fields for Register: HFLBAddr**

Bits	Name	Memory Access	Description
31:0	HFLBAddr	R/W	<p>The starting address of the Frame list. This register is used only for Isochronous and Interrupt Channels.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.39 HPRT

- **Name:** Host Port Control and Status Register
- **Description:** This register is available only in Host mode. Currently, the OTG Host supports only one port. A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in the "Interrupt Hierarchy" figure in the databook. The R\_SS\_WC bits in this register can trigger an interrupt to the application through the Host Port Interrupt bit of the Core Interrupt register (GINTSTS.PrtInt). On a Port Interrupt, the application must read this register and clear the bit that caused the interrupt. For the R\_SS\_WC bits, the application must write a 1 to the bit to clear the interrupt.
- **Size:** 32 bits
- **Offset:** 0x440
- **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==5 || OTG\_MODE == 6

Rsvd	31:19	18:17	16:13	12	11:10	RESERVED	9	8	7	6	5	PrtOvrCurrChng	4	3	2	1	0	
PrtSpd			PrtTstCtl	PrtPwr	PrtLnsts		PrtRst	PrtSusp	PrtRes			PrtOvrCurrAct		PrtEnChng		PrtEna	PrtConnDet	

**Table 7-44 Fields for Register: HPRT**

Bits	Name	Memory Access	Description
31:19			<b>Reserved Field:</b> Yes

**Table 7-44 Fields for Register: HPRT (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
18:17	PrtSpd	R	<p>Port Speed (PrtSpd)      Indicates the speed of the device attached to this port.</p> <ul style="list-style-type: none"> <li>■ 2'b00: High speed</li> <li>■ 2'b01: Full speed</li> <li>■ 2'b10: Low speed</li> <li>■ 2'b11: Reserved</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (HIGHSPD): High speed</li> <li>■ 0x1 (FULLSPD): Full speed</li> <li>■ 0x2 (LOWSPD): Low speed</li> <li>■ 0x3 (RESERVED): Reserved</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-44 Fields for Register: HPRT (Continued)**

Bits	Name	Memory Access	Description
16:13	PrTstCtl	R/W	<p>Port Test Control (PrTstCtl)  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> <ul style="list-style-type: none"> <li>■ 4'b0000: Test mode disabled</li> <li>■ 4'b0001: Test_J mode</li> <li>■ 4'b0010: Test_K mode</li> <li>■ 4'b0011: Test_SE0_NAK mode</li> <li>■ 4'b0100: Test_Packet mode</li> <li>■ 4'b0101: Test_Force_Enable</li> <li>■ Others: Reserved</li> </ul> <p>To move the DWC_otg controller to test mode, you must set this field. Complete the following steps to move the DWC_otg core to test mode:</p> <ul style="list-style-type: none"> <li>■ 1. Power on the core.</li> <li>■ 2. Load the DWC_otg driver.</li> <li>■ 3. Connect an HS device and enumerate to HS mode.</li> <li>■ 4. Access the HPRT register to send test packets.</li> <li>■ 5. Remove the device and connect to fixture (OPT) port. The DWC_otg host core continues sending out test packets.</li> <li>■ 6. Test the eye diagram.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Test mode disabled</li> <li>■ 0x1 (TESTJ): Test_J mode</li> <li>■ 0x2 (TESTK): Test_K mode</li> <li>■ 0x3 (TESTSN): Test_SE0_NAK mode</li> <li>■ 0x4 (TESTPM): Test_Packet mode</li> <li>■ 0x5 (TESTFENB): Test_force_Enable</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-44 Fields for Register: HPRT (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
12	PrtPwr	R/W	<p>Port Power (PrtPwr)  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> <ul style="list-style-type: none"> <li>■ 1'b0: Power off</li> <li>■ 1'b1: Power on</li> </ul> <p><b>Note:</b> This bit is interface independent. The application needs to program this bit for all interfaces as described in the host programming flow in the Programming Guide.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (OFF): Power off</li> <li>■ 0x1 (ON): Power on</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
11:10	PrtLnSts	R	<p>Port Line Status (PrtLnSts)  Indicates the current logic level USB data lines</p> <ul style="list-style-type: none"> <li>■ Bit [10]: Logic level of D+</li> <li>■ Bit [11]: Logic level of D-</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (PLUSD): Logic level of D+</li> <li>■ 0x2 (MINUSD): Logic level of D-</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>
9	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-44 Fields for Register: HPRT (Continued)**

Bits	Name	Memory Access	Description
8	PrtRst	R/W	<p>Port Reset (PrtRst)</p> <p>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> <ul style="list-style-type: none"> <li>■ 1'b0: Port not in reset</li> <li>■ 1'b1: Port in reset</li> </ul> <p>The application must leave this bit set for at least a minimum duration mentioned below to start a reset on the port. 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. This bit is cleared by the core even if there is no device connected to the Host.</p> <ul style="list-style-type: none"> <li>■ High speed: 50 ms</li> <li>■ Full speed/Low speed: 10 ms</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Port not in reset</li> <li>■ 0x1 (ENABLED): Port in reset</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-44 Fields for Register: HPRT (Continued)**

Bits	Name	Memory Access	Description
7	PrtSusp	R/W1S	<p>Port Suspend (PrtSusp)  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). This bit is cleared by the core even if there is no device connected to the Host.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Port not in Suspend mode</li> <li>■ 1'b1: Port in Suspend mode</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Port not in Suspend mode</li> <li>■ 0x1 (ACTIVE): Port in Suspend mode</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>

**Table 7-44 Fields for Register: HPRT (Continued)**

Bits	Name	Memory Access	Description
6	PrtRes	R/W	<p>Port Resume (PrtRes)  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> <ul style="list-style-type: none"> <li>■ 1'b0: No resume driven</li> <li>■ 1'b1: Resume driven</li> </ul> <p>When LPM is enabled, In L1 state the behavior of this bit is as follows: The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until a pre-determined time specified in GLPMCFG.HIRD_Thres[3:0] field. If the core detects a USB remote wakeup sequence, as indicated by the Port L1Resume/Remote L1Wakeup 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 resume. This bit can be set by both core or application and also cleared by core or application. This bit is cleared by the core even if there is no device connected to the Host.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NORESUME): No resume driven</li> <li>■ 0x1 (RESUME): Resume driven</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>
5	PrtOvrCurrChng	R/W1C	<p>Port Overcurrent Change (PrtOvrCurrChng)  The core sets this bit when the status of the Port Overcurrent Active bit (bit 4) in this register changes. This bit can be set only by the core and the application should write 1 to clear it</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Status of port overcurrent status is not changed</li> <li>■ 0x1 (ACTIVE): Status of port overcurrent changed</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-44 Fields for Register: HPRT (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
4	PrtOvrCurrAct	R	<p>Port Overcurrent Active (PrtOvrCurrAct) Indicates the overcurrent condition of the port.</p> <ul style="list-style-type: none"> <li>■ 1'b0: No overcurrent condition</li> <li>■ 1'b1: Overcurrent condition</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No overcurrent condition</li> <li>■ 0x1 (ACTIVE): Overcurrent condition</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
3	PrtEnChng	R/W1C	<p>Port Enable/Disable Change (PrtEnChng) The core sets this bit when the status of the Port Enable bit [2] of this register changes. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Port Enable bit 2 has not changed</li> <li>■ 0x1 (ACTIVE): Port Enable bit 2 changed</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
2	PrtEna	R/W1C	<p>Port Enable (PrtEna) 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 by writing 1. This bit does not trigger any interrupt to the application.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Port disabled</li> <li>■ 1'b1: Port enabled</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Port disabled</li> <li>■ 0x1 (ENABLED): Port enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-44 Fields for Register: HPRT (Continued)**

Bits	Name	Memory Access	Description
1	PrtConnDet	R/W1C	<p>Port Connect Detected (PrtConnDet)  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.Prtlnt). This bit can be set only by the core and the application should write 1 to clear it. The application must write a 1 to this bit to clear the interrupt.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (ACTIVE): Device connection detected</li> <li>■ 0x0 (INACTIVE): No device connection detected</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
0	PrtConnSts	R	<p>Port Connect Status (PrtConnSts)</p> <ul style="list-style-type: none"> <li>■ 0: No device is attached to the port.</li> <li>■ 1: A device is attached to the port.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTATTACHED): No device is attached to the port</li> <li>■ 0x1 (ATTACHED): A device is attached to the port</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.40 HCCHARi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)

- **Name:** Host Channel Characteristics Register
- **Description:** This register contains the characteristics of the Host Channel.
- **Size:** 32 bits
- **Offset:** 0x500 + i\*20
- **Exists:** (OTG\_MODE !=3) && (OTG\_MODE !=4) && (0 < OTG\_NUM\_HOST\_CHAN)

ChEna	31	ChDis	30	OddFrm	29	DevAddr	28:22	EC	21:20	EPType	19:18	LSpdDev	17	RESERVED	16	EPDir	15	EPNum	14:11	MPS	10:0
-------	----	-------	----	--------	----	---------	-------	----	-------	--------	-------	---------	----	----------	----	-------	----	-------	-------	-----	------

**Table 7-45 Fields for Register: HCCHARi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
31	ChEna	R/W1S	<p>Channel Enable (ChEna) When Scatter/Gather mode is enabled</p> <ul style="list-style-type: none"> <li>■ 1'b0: Indicates that the descriptor structure is not yet ready.</li> <li>■ 1'b1: Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor.</li> </ul> <p>When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Channel disabled</li> <li>■ 1'b1: Channel enabled</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): If Scatter/Gather mode is enabled, indicates that the descriptor structure is not yet ready. If Scatter/Gather mode is disabled, indicates that the channel is disabled.</li> <li>■ 0x1 (ENABLED): If Scatter/Gather mode is enabled, indicates that the descriptor structure and data buffer with data is set up and this channel can access the descriptor. If Scatter/Gather mode is disabled, indicates that the channel is enabled.</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
30	ChDis	R/W1S	<p>Channel Disable (ChDis) 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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Transmit/Recieve normal</li> <li>■ 0x1 (ACTIVE): Stop transmitting/receiving data on channel</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

**Table 7-45 Fields for Register: HCCHAR<sub>i</sub> (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
29	OddFrm	R/W	<p>Odd Frame (OddFrm)  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> <ul style="list-style-type: none"> <li>■ 1'b0: Even (micro)Frame</li> <li>■ 1'b1: Odd (micro)Frame</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (EFRAME): Even Frame Transfer</li> <li>■ 0x1 (OFRAME): Odd Frame Transfer</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
28:22	DevAddr	R/W	<p>Device Address (DevAddr)  This field selects the specific device serving as the data source or sink.</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-45 Fields for Register: HCCHARi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
21:20	EC	R/W	<p>Multi Count (MC) / Error Count (EC)  When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SplitEna) 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> <ul style="list-style-type: none"> <li>■ 2'b00: Reserved This field yields undefined results.</li> <li>■ 2'b01: 1 transaction</li> <li>■ 2'b10: 2 transactions to be issued for this endpoint per microframe</li> <li>■ 2'b11: 3 transactions to be issued for this endpoint per microframe</li> </ul> <p>When HCSPLTn.SplitEna 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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (RESERVED): Reserved. This field yields undefined result</li> <li>■ 0x1 (TRANSONE): 1 transaction</li> <li>■ 0x2 (TRANSTWO): 2 transactions to be issued for this endpoint per microframe</li> <li>■ 0x3 (TRANSTHREE): 3 transactions to be issued for this endpoint per microframe</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-45 Fields for Register: HCCHARi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

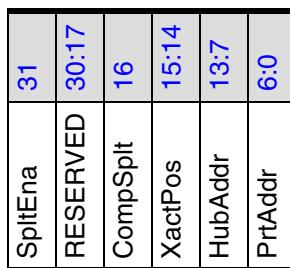
Bits	Name	Memory Access	Description
19:18	EPType	R/W	<p>Endpoint Type (EPType) Indicates the transfer type selected.</p> <ul style="list-style-type: none"> <li>■ 2'b00: Control</li> <li>■ 2'b01: Isochronous</li> <li>■ 2'b10: Bulk</li> <li>■ 2'b11: Interrupt</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (CTRL): Control</li> <li>■ 0x1 (ISOC): Isochronous</li> <li>■ 0x2 (BULK): Bulk</li> <li>■ 0x3 (INTERR): Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
17	LSpdDev	R/W	<p>Low-Speed Device (LSpdDev) This field is Set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The DWC_otg Host core uses this field to drive the XCVR_SELECT signal to 2'b11 while communicating to the LS Device through the FS hub. <b>Note:</b> In a peer to peer setup, the DWC_otg Host core ignores this bit even if it is set by the application software.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Not Communicating with low speed device</li> <li>■ 0x1 (ENABLED): Communicating with low speed device</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
16	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead</p>

**Table 7-45 Fields for Register: HCCHARi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
15	EPDir	R/W	<p>Endpoint Direction (EPDir) Indicates whether the transaction is IN or OUT.</p> <ul style="list-style-type: none"> <li>■ 1'b0: OUT</li> <li>■ 1'b1: IN</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (OUT): OUT Direction</li> <li>■ 0x1 (IN): IN Direction</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
14:11	EPNum	R/W	<p>Endpoint Number (EPNum) Indicates the endpoint number on the device serving as the data source or sink.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ENDPT0): End point 0</li> <li>■ 0x1 (ENDPT1): End point 1</li> <li>■ 0x2 (ENDPT2): End point 2</li> <li>■ 0x3 (ENDPT3): End point 3</li> <li>■ 0x4 (ENDPT4): End point 4</li> <li>■ 0x5 (ENDPT5): End point 5</li> <li>■ 0x6 (ENDPT6): End point 6</li> <li>■ 0x7 (ENDPT7): End point 7</li> <li>■ 0x8 (ENDPT8): End point 8</li> <li>■ 0x9 (ENDPT9): End point 9</li> <li>■ 0xa (ENDPT10): End point 10</li> <li>■ 0xb (ENDPT11): End point 11</li> <li>■ 0xc (ENDPT12): End point 12</li> <li>■ 0xd (ENDPT13): End point 13</li> <li>■ 0xe (ENDPT14): End point 14</li> <li>■ 0xf (ENDPT15): End point 15</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
10:0	MPS	R/W	<p>Maximum Packet Size (MPS) Indicates the maximum packet size of the associated endpoint.</p> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

### 7.1.41 HCSPLTi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)

- **Name:** Host Channel Split Control Register
- **Description:** This register contains the Split characteristics of the Host Channel.
- **Size:** 32 bits
- **Offset:** 0x504 + i\*20
- **Exists:** (OTG\_MODE !=3) && (OTG\_MODE !=4) && OTG\_SINGLE\_POINT == 0 && (OTG\_NUM\_HOST\_CHAN > 0)



**Table 7-46 Fields for Register: HCSPLTi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)**

Bits	Name	Memory Access	Description
31	SplitEna	R/W	<p>Split Enable (SplitEna) The application sets this field to indicate that this channel is enabled to perform split transactions.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Split not enabled</li> <li>■ 0x1 (ENABLED): Split enabled</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
30:17	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead</p>

**Table 7-46 Fields for Register: HCSPLTi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

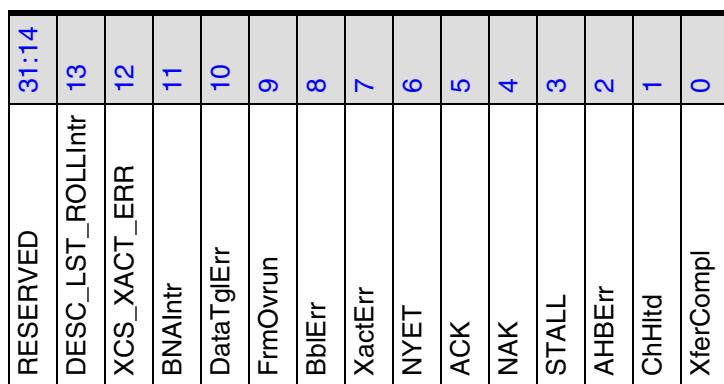
Bits	Name	Memory Access	Description
16	CompSplt	R/W	<p>Do Complete Split (CompSplt) The application sets this field to request the OTG host to perform a complete split transaction.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOSPLIT): No complete split transaction</li> <li>■ 0x1 (SPLIT): Complete Split transaction</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
15:14	XactPos	R/W	<p>Transaction Position (XactPos) This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction.</p> <ul style="list-style-type: none"> <li>■ 2'b11: All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes).</li> <li>■ 2'b10: Begin. This is the first data payload of this transaction (which is larger than 188 bytes).</li> <li>■ 2'b00: Mid. This is the middle payload of this transaction (which is larger than 188 bytes).</li> <li>■ 2'b01: End. This is the last payload of this transaction (which is larger than 188 bytes).</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MIDDLE): Mid. This is the middle payload of this transaction (which is larger than 188 bytes)</li> <li>■ 0x1 (END): End. This is the last payload of this transaction (which is larger than 188 bytes)</li> <li>■ 0x2 (BEGIN): Begin. This is the first data payload of this transaction (which is larger than 188 bytes)</li> <li>■ 0x3 (ALL): All. This is the entire data payload of this transaction (which is less than or equal to 188 bytes)</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

**Table 7-46 Fields for Register: HCSPLTi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
13:7	HubAddr	R/W	<p>Hub Address (HubAddr)            This field holds the device address of the transaction translator's hub.</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
6:0	PrtAddr	R/W	<p>Port Address (PrtAddr)            This field is the port number of the recipient transaction translator.</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

### 7.1.42 HCINT<sub>i</sub> (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)

- **Name:** Host Channel Interrupt Register
- **Description:** This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in the "Interrupt Hierarchy" figure in the databook. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAIINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAIINT and GINTSTS registers.
- **Size:** 32 bits
- **Offset:** 0x508 + i\*20
- **Exists:** (OTG\_MODE !=3) && (OTG\_MODE !=4) && (OTG\_NUM\_HOST\_CHAN > 0)



**Table 7-47 Fields for Register: HCINT<sub>i</sub> (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)**

Bits	Name	Memory Access	Description
31:14	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-47 Fields for Register: HCINT<sub>i</sub> (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
13	DESC_LST_ROLLIntr	R/W1C	<p>Descriptor rollover interrupt (DESC_LST_ROLLIntr)  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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Descriptor rollover interrupt</li> <li>■ 0x1 (ACTIVE): Descriptor rollover interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA == 1</p>
12	XCS_XACT_ERR	R/W1C	<p>Excessive Transaction Error (XCS_XACT_ERR)  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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Excessive Transaction Error</li> <li>■ 0x1 (ACTIVE): Excessive Transaction Error</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA == 1</p>
11	BNAIntr	R/W1C	<p>BNA (Buffer Not Available) Interrupt (BNAIntr)  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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No BNA Interrupt</li> <li>■ 0x1 (ACTIVE): BNA Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA == 1</p>
10	DataTglErr	R/W1C	<p>Data Toggle Error (DataTglErr). This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Data Toggle Error</li> <li>■ 0x1 (ACTIVE): Data Toggle Error</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-47 Fields for Register: HCINT<sub>i</sub> (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
9	FrmOvrn	R/W1C	<p>Frame Overrun (FrmOvrn). In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Frame Overrun</li> <li>■ 0x1 (ACTIVE): Frame Overrun</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
8	BblErr	R/W1C	<p>Babble Error (BblErr) In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Babble Error</li> <li>■ 0x1 (ACTIVE): Babble Error</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
7	XactErr	R/W1C	<p>Transaction Error (XactErr) Indicates one of the following errors occurred on the USB.</p> <ul style="list-style-type: none"> <li>■ CRC check failure</li> <li>■ Timeout</li> <li>■ Bit stuff error</li> <li>■ False EOP</li> </ul> <p>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Transaction Error</li> <li>■ 0x1 (ACTIVE): Transaction Error</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-47 Fields for Register: HCINT<sub>i</sub> (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
6	NYET	R/W1C	<p>NYET Response Received Interrupt (NYET)            In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No NYET Response Received Interrupt</li> <li>■ 0x1 (ACTIVE): NYET Response Received Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
5	ACK	R/W1C	<p>ACK Response Received/Transmitted Interrupt (ACK)            In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No ACK Response Received or Transmitted Interrupt</li> <li>■ 0x1 (ACTIVE): ACK Response Received or Transmitted Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
4	NAK	R/W1C	<p>NAK Response Received Interrupt (NAK)            In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No NAK Response Received Interrupt</li> <li>■ 0x1 (ACTIVE): NAK Response Received Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
3	STALL	R/W1C	<p>STALL Response Received Interrupt (STALL)            In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Stall Response Received Interrupt</li> <li>■ 0x1 (ACTIVE): Stall Response Received Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-47 Fields for Register: HCINT<sub>i</sub> (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
2	AHBErr	R/W1C	<p>AHB Error (AHBErr)  This is generated only in Internal 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. For details, see "AHB Error Handling" in the Programming Guide.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No AHB error</li> <li>■ 0x1 (ACTIVE): AHB error during AHB read/write</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE == 2</p>
1	ChHlt <sub>d</sub>	R/W1C	<p>Channel Halted (ChHlt<sub>d</sub>)  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</p> <ul style="list-style-type: none"> <li>■ EOL being set in descriptor</li> <li>■ AHB error</li> <li>■ Excessive transaction errors</li> <li>■ Babble</li> <li>■ Stall</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Channel not halted</li> <li>■ 0x1 (ACTIVE): Channel Halted</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-47 Fields for Register: HCINT<sub>i</sub> (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
0	XferCompl	R/W1C	<p>Transfer Completed (XferCompl)            Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it.</p> <ul style="list-style-type: none"> <li>■ For Scatter/Gather DMA mode, it indicates that current descriptor processing got completed with IOC bit set in its descriptor.</li> <li>■ In non Scatter/Gather DMA mode, it indicates that Transfer completed normally without any errors.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Transfer in progress or No Active Transfer</li> <li>■ 0x1 (ACTIVE): Transfer completed normally without any errors</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.43 HCINTMSKi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)

- **Name:** Host Channel Interrupt Mask Register
- **Description:** This register reflects the mask for each channel status described in the previous section.
- **Size:** 32 bits
- **Offset:** 0x50C + i\*20
- **Exists:** (OTG\_MODE !=3) && (OTG\_MODE !=4) && (OTG\_NUM\_HOST\_CHAN > 0)

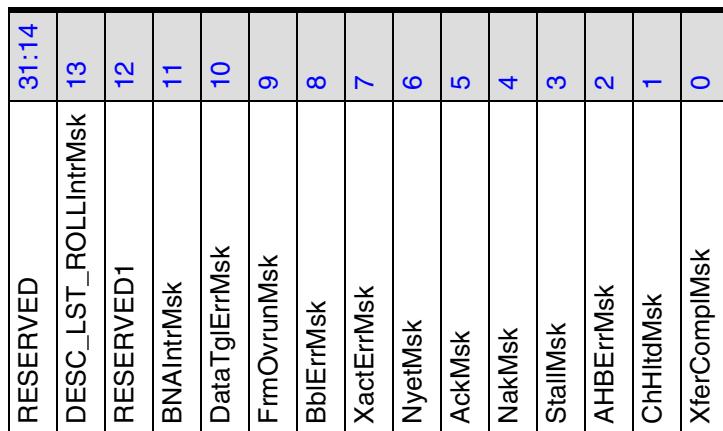


Table 7-48 Fields for Register: HCINTMSKi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)

Bits	Name	Memory Access	Description
31:14	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
13	DESC_LST_ROLLIntrMsk	R/W	<p>Descriptor List rollover interrupt Mask register(DESC_LST_ROLLIntrMsk) This bit is valid only when Scatter/Gather DMA mode is enabled.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Descriptor Rollover Interrupt Mask</li> <li>■ 0x1 (NOMASK): Descriptor Rollover Interrupt not masked</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA == 1</p> <p><b>Testable:</b> readOnly</p>

**Table 7-48 Fields for Register: HCINTMSKi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
12	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
11	BNAIntrMsk	R/W	<p>BNA (Buffer Not Available) Interrupt mask register (BNAIntrMsk) This bit is valid only when Scatter/Gather DMA mode is enabled.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): BNA Interrupt Masked</li> <li>■ 0x1 (NOMASK): BNA Interrupt not masked</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA == 1</p> <p><b>Testable:</b> readOnly</p>
10	DataTglErrMsk	R/W	<p>Data Toggle Error Mask (DataTglErrMsk) In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Data Toggle Error</li> <li>■ 0x1 (NOMASK): No Data Toggle Error Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA != 1</p>
9	FrmOvrnMsk	R/W	<p>Frame Overrun Mask (FrmOvrnMsk) In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Overrun Mask</li> <li>■ 0x1 (NOMASK): No Frame Overrun Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA != 1</p>
8	BblErrMsk	R/W	<p>Babble Error Mask (BblErrMsk) In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Babble Error</li> <li>■ 0x1 (NOMASK): No Babble Error Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA != 1</p>

**Table 7-48 Fields for Register: HCINTMSKi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

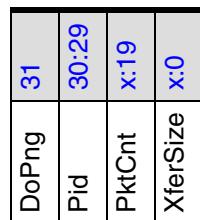
Bits	Name	Memory Access	Description
7	XactErrMsk	R/W	<p>Transaction Error Mask (XactErrMsk) In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Transaction Error</li> <li>■ 0x1 (NOMASK): No Transaction Error Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA != 1</p>
6	NyetMsk	R/W	<p>NYET Response Received Interrupt Mask (NyettMsk) In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask NYET Response Received Interrupt</li> <li>■ 0x1 (NOMASK): No NYET Response Received Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA != 1</p>
5	AckMsk	R/W	<p>ACK Response Received/Transmitted Interrupt Mask (AckMsk) In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask ACK Response Received/Transmitted Interrupt</li> <li>■ 0x1 (NOMASK): No ACK Response Received/Transmitted Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA != 1</p>
4	NakMsk	R/W	<p>NAK Response Received Interrupt Mask (NakMsk) In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask NAK Response Received Interrupt</li> <li>■ 0x1 (NOMASK): No NAK Response Received Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA != 1</p>

**Table 7-48 Fields for Register: HCINTMSKi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
3	StallMsk	R/W	<p>STALL Response Received Interrupt Mask (StallMsk) In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask STALL Response Received Interrupt</li> <li>■ 0x1 (NOMASK): No STALL Response Received Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA != 1</p>
2	AHBErrMsk	R/W	<p>AHB Error Mask (AHBErrMsk) In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): AHB Error Mask</li> <li>■ 0x1 (NOMASK): No AHB Error Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE == 2</p>
1	ChHltdMsk	R/W	<p>Channel Halted Mask (ChHltdMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Channel Halted Mask</li> <li>■ 0x1 (NOMASK): No Channel Halted Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
0	XferComplMsk	R/W	<p>Transfer Completed Mask (XferComplMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Transfer Completed Mask</li> <li>■ 0x1 (NOMASK): No Transfer Completed Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.44 HCTSIZi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)

- **Name:** Host Channel Transfer Size Register
- **Description:** This register reflects the transfer size for the Host Channel.
- **Size:** 32 bits
- **Offset:** 0x510 + i\*20
- **Exists:** (OTG\_MODE !=3) && (OTG\_MODE !=4) && (OTG\_NUM\_HOST\_CHAN > 0)



**Table 7-49 Fields for Register: HCTSIZi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)**

Bits	Name	Memory Access	Description
31	DoPng	R/W	<p>Do Ping (DoPng) This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol. <b>Note:</b> Do not set this bit for IN transfers. If this bit is set for IN transfers it disables the channel.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOPING): No ping protocol</li> <li>■ 0x1 (PING): Ping protocol</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

**Table 7-49 Fields for Register: HCTSIZi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
30:29	Pid	R/W	<p>PID (Pid)            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> <ul style="list-style-type: none"> <li>■ 2'b00: DATA0</li> <li>■ 2'b01: DATA2</li> <li>■ 2'b10: DATA1</li> <li>■ 2'b11: MDATA (non-control)/SETUP (control)</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DATA0): DATA0</li> <li>■ 0x1 (DATA2): DATA2</li> <li>■ 0x2 (DATA1): DATA1</li> <li>■ 0x3 (MDATA): MDATA (non-control)/SETUP (control)</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
x:19	PktCnt	R/W	<p><b>Non-Scatter/Gather DMA Mode:</b>            Packet Count (PktCnt)            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 during coreConsultant configuration (parameter OTG_PACKET_COUNT_WIDTH).</p> <p><b>Scatter/Gather DMA Mode:</b>  [28:19]: Reserved</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always  <b>Range Variable[x]:</b> OTG_PACKET_COUNT_WIDTH + 18</p>

**Table 7-49 Fields for Register: HCTSIZ*i* (for *i* = 0; *i* <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
x:0	XferSize	R/W	<p><b>Non-Scatter/Gather DMA Mode:</b>  Transfer Size (XferSize)  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 during coreConsultant configuration (parameter OTG_TRANS_COUNT_WIDTH).</p> <p><b>Scatter/Gather DMA Mode:</b>  [18:16]: Reserved  [15:8]: NTD (Number of Transfer Descriptors)  <i>(Non Isochronous)</i>  This value is in terms of number of descriptors. Maximum number of descriptor that can be present in the list is 64. The values can be from 0 to 63.</p> <ul style="list-style-type: none"> <li>■ 0: 1 descriptor</li> <li>■ 63: 64 descriptors</li> </ul> <p>This field indicates the total number of descriptors present in that list. The core will wrap around after servicing NTD number of descriptors for that list.  <i>(Isochronous)</i>  This field indicates the number of descriptors present in that list microframe.  The possible values for FS are</p> <ul style="list-style-type: none"> <li>■ 1: 2 descriptors</li> <li>■ 3: 4 descriptors</li> <li>■ 7: 8 descriptors</li> <li>■ 15: 16 descriptors</li> <li>■ 31: 32 descriptors</li> <li>■ 63: 64 descriptors</li> </ul> <p>The possible values for HS are</p> <ul style="list-style-type: none"> <li>■ 7: 8 descriptors</li> <li>■ 15: 16 descriptors</li> <li>■ 31: 32 descriptors</li> <li>■ 63: 64 descriptors</li> <li>■ 127: 128 descriptors</li> <li>■ 255: 256 descriptors</li> </ul> <p>[7:0]: SCHED_INFO (Schedule information)</p>

**Table 7-49 Fields for Register: HCTSIZi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
			<p>Every bit in this 8 bit register indicates scheduling for that microframe.</p> <p>Bit 0 indicates scheduling for 1st microframe and bit 7 indicates scheduling for 8th microframe in that frame.</p> <p>A value of 8'b11111111 indicates that the corresponding interrupt channel is scheduled to issue a token every microframe in that frame. A value of 8'b10101010 indicates that the corresponding interrupt channel is scheduled to issue a token every alternate microframe starting with second microframe. Note that this field is applicable only for periodic (Isochronous and Interrupt) channels.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Range Variable[x]:</b> OTG_TRANS_COUNT_WIDTH - 1</p>

### 7.1.45 HCDMAi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)

- **Name:** Host Channel DMA Address Register
- **Description:** This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.
- **Size:** 32 bits
- **Offset:** 0x514 + i\*20
- **Exists:** (OTG\_MODE !=3) && (OTG\_MODE !=4) && (OTG\_NUM\_HOST\_CHAN > 0) && (OTG\_ARCHITECTURE == 2)



**Table 7-50 Fields for Register: HCDMAi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)**

Bits	Name	Memory Access	Description
31:0	DMAAddr	R/W	<p><b>In Buffer DMA Mode:</b>  [31:0]: DMA Address (DMAAddr)  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.  <b>Reset:</b> X if not programmed as the register is in SPRAM.</p> <p><b>In Scatter-Gather DMA (DescDMA) Mode for Non-Isochronous:</b>  [31:9]: DMA Address (DMAAddr)  The start address must be 512-bytes aligned.  This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value.  [8:3]: Current Transfer Desc(CTD)  This value is in terms of number of descriptors. The values can be from 0 to 63. <ul style="list-style-type: none"> <li>■ 0 - 1 descriptor.</li> <li>■ 63 - 64 descriptors.</li> </ul> This field indicates the current descriptor processed in the list. This field is updated both by application and the core. For example, if the application enables the channel after programming CTD=5, then the core will start processing the sixth descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr.  <b>Reset:</b> 6'h0  [2:0]: Reserved</p> <p><b>In Scatter-Gather DMA (DescDMA) Mode for Isochronous:</b>  [31:N]: DMA Address (DMAAddr)  The start address must be 512-bytes aligned.  This field holds the address of the <math>2^*(nTD+1)</math> bytes of locations in which the isochronous descriptors are present where N is based on nTD as follows: <ul style="list-style-type: none"> <li>■ [31:N]: Base Address</li> <li>■ [N-1:3]: Offset</li> <li>■ [2:0]: 000</li> </ul> For HS ISOC, if nTD is, <ul style="list-style-type: none"> <li>■ 7, N=6</li> <li>■ 15, N=7</li> </ul> </p>

**Table 7-50 Fields for Register: HCDMAi (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1) (Continued)**

Bits	Name	Memory Access	Description
			<ul style="list-style-type: none"> <li>■ 31, N=8</li> <li>■ 63, N=9</li> <li>■ 127, N=10</li> <li>■ 255, N=11</li> </ul> <p>For FS ISOC, if nTD is,</p> <ul style="list-style-type: none"> <li>■ 1, N=4</li> <li>■ 3, N=5</li> <li>■ 7, N=6</li> <li>■ 15, N=7</li> <li>■ 31, N=8</li> <li>■ 63, N=9</li> </ul> <p><i>[N-1:3]: Current Transfer Desc(CTD)</i> CTD for isochronous is based on the current frame/(micro)frame value. Need to be set to zero by application.</p> <p><b>Reset:</b> (N+1:3)'h0  <b>[2:0]: Reserved</b>  <b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

### 7.1.46 HCDMAB<sub>i</sub> (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)

- **Name:** Host Channel DMA Buffer Address Register
- **Description:** This register is present only in case of Scatter/Gather DMA. It is implemented in RAM instead of flop-based implementation. This register holds the current buffer address.
- **Size:** 32 bits
- **Offset:** 0x51C + i\*20
- **Exists:** (OTG\_MODE !=3) && (OTG\_MODE !=4) && (OTG\_NUM\_HOST\_CHAN > 0) && OTG\_EN\_DESC\_DMA == 1



**Table 7-51 Fields for Register: HCDMAB<sub>i</sub> (for i = 0; i <= OTG\_NUM\_HOST\_CHAN-1)**

Bits	Name	Memory Access	Description
31:0	HCDMAB	R	<p>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> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> untestable</p>

### 7.1.47 DCFG

- **Name:** Device Configuration Register
- **Description:** This register configures the core in Device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.
- **Size:** 32 bits
- **Offset:** 0x800
- **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==3 || OTG\_MODE == 4

ResValid	31:26
PerSchlntvl	25:24
DescDMA	23
EPMisCnt	22:18
ipgisocSupt	17
RESERVED	16
EraticIntMsk	15
XCVRDLY	14
EnDevOutNak	13
PerFrint	12:11
DevAddr	10:4
Ena32KHzSusp	3
NZStsOUTHShk	2
DevSpd	1:0

**Table 7-52 Fields for Register: DCFG**

Bits	Name	Memory Access	Description
31:26	ResValid	R/W	<p>Resume Validation Period (ResValid)            This field is effective only when DCFG.Ena32KHzSusp is set. It controls the resume period when the core resumes from suspend. The core counts for ResValid number of clock cycles to detect a valid resume when this bit is set  <b>Value After Reset:</b> 0x2  <b>Exists:</b> Always</p>

**Table 7-52 Fields for Register: DCFG (Continued)**

Bits	Name	Memory Access	Description
25:24	PerSchIntvl	R/W	<p>Periodic Scheduling Interval (PerSchIntvl)      PerSchIntvl must be programmed for Scatter/Gather DMA mode.      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.</p> <ul style="list-style-type: none"> <li>■ When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data .</li> <li>■ When no periodic endpoints are active, Then the internal DMA engine services non-periodic endpoints, ignoring this field.</li> <li>■ After the specified time within a (micro)Frame, the DMA switches to fetching for non-periodic endpoints.             <ul style="list-style-type: none"> <li>- 2'b00: 25% of (micro)Frame.</li> <li>- 2'b01: 50% of (micro)Frame.</li> <li>- 2'b10: 75% of (micro)Frame.</li> <li>- 2'b11: Reserved.</li> </ul> </li> </ul> <p>Reset: 2'b00</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MF25): 25% of (micro)Frame</li> <li>■ 0x1 (MF50): 50% of (micro)Frame</li> <li>■ 0x2 (MF75): 75% of (micro)Frame</li> <li>■ 0x3 (RESERVED): Reserved</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-52 Fields for Register: DCFG (Continued)**

Bits	Name	Memory Access	Description
23	DescDMA	R/W	<p>Enable Scatter/gather DMA in device mode (DescDMA). 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.</p> <p>Note: This bit must be modified only once after a reset. The following combinations are available for programming:</p> <ul style="list-style-type: none"> <li>■ GAHBCFG.DMAEn=0, DCFG.DescDMA=0 =&gt; Slave mode</li> <li>■ GAHBCFG.DMAEn=0, DCFG.DescDMA=1 =&gt; Invalid</li> <li>■ GAHBCFG.DMAEn=1, DCFG.DescDMA=0 =&gt; Buffered DMA mode</li> <li>■ GAHBCFG.DMAEn=1, DCFG.DescDMA=1 =&gt; Scatter/Gather DMA mode</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disable Scatter/Gather DMA</li> <li>■ 0x1 (ENABLED): Enable Scatter/Gather DMA</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA == 1</p>
22:18	EPMisCnt	R/W	<p>IN Endpoint Mismatch Count (EPMisCnt)</p> <p>This field is valid only in shared FIFO operation.</p> <p>The application programs this feild 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> <p><b>Value After Reset:</b> 0x8</p> <p><b>Exists:</b> OTG_EN_DED_TX_FIFO ==0</p>

**Table 7-52 Fields for Register: DCFG (Continued)**

Bits	Name	Memory Access	Description
17	ipgisocSupt	R/W	<p>Worst-Case Inter-Packet Gap ISOC OUT Support (ipgisocSupt)  This bit indicates that the controller supports the worst-case scenario of Rx followed by Rx Inter-Packet Gap (IPG) (32 bit times) as per the <i>UTMI Specification</i> for any token following an ISOC OUT token. Without this support, when any token follows an ISOC OUT token with the worst-case IPG, the controller will not detect the followed token. The worst-case IPG of the controller without this support depends on the AHB and PHY clock frequency.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Worst-Case Inter-Packet Gap ISOC OUT Support is disabled</li> <li>■ 0x1 (ENABLED): Worst-Case Inter-Packet Gap ISOC OUT Support is enabled</li> </ul> <p><b>Value After Reset:</b> 0x1  <b>Exists:</b> OTG_ISOC_IPG_ENH ==1</p>
16	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> 0  <b>Testable:</b> writeAsRead  <b>Write Constraint:</b> writeAsRead</p>
15	ErraticIntMsk	R/W	<p>Erratic Error Interrupt Mask</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOMASK): Early suspend interrupt is generated on erratic error</li> <li>■ 0x1 (MASK): Mask early suspend interrupt on erratic error</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
14	XCVRDLY	R/W	<p>XCVRDLY</p> <p>Enables or disables delay between xcvr_sel and txvalid during device chirp</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): No delay between xcvr_sel and txvalid during Device chirp</li> <li>■ 0x1 (ENABLE): Enable delay between xcvr_sel and txvalid during Device chirp</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-52 Fields for Register: DCFG (Continued)**

Bits	Name	Memory Access	Description
13	EnDevOutNak	R/W	<p>Enable Device OUT NAK (EnDevOutNak)  This bit enables setting NAK for Bulk OUT endpoints after the transfer is completed for Device mode Descriptor DMA</p> <ul style="list-style-type: none"> <li>■ 1'b0 : The core does not set NAK after Bulk OUT transfer complete</li> <li>■ 1'b1 : The core sets NAK after Bulk OUT transfer complete</li> </ul> <p>This bit is one time programmable after reset like any other DCFG register bits.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED):  The core does not set NAK after Bulk OUT transfer complete</li> <li>■ 0x1 (ENABLED): The core sets NAK after Bulk OUT transfer complete</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_DESC_DMA == 1</p>
12:11	PerFrInt	R/W	<p>Periodic Frame Interval (PerFrInt)  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.</p> <ul style="list-style-type: none"> <li>■ 2'b00: 80% of the (micro)Frame interval</li> <li>■ 2'b01: 85% of the (micro)Frame interval</li> <li>■ 2'b10: 90% of the (micro)Frame interval</li> <li>■ 2'b11: 95% of the (micro)Frame interval</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (EOPF80): 80% of the (micro)Frame interval</li> <li>■ 0x1 (EOPF85): 85% of the (micro)Frame interval</li> <li>■ 0x2 (EOPF90): 90% of the (micro)Frame interval</li> <li>■ 0x3 (EOPF95): 95% of the (micro)Frame interval</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
10:4	DevAddr	R/W	<p>Device Address (DevAddr)  The application must program this field after every SetAddress control command.</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-52 Fields for Register: DCFG (Continued)**

Bits	Name	Memory Access	Description
3	Ena32KHzSusp	R/W	<p>Enable 32 KHz Suspend mode (Ena32KHzSusp)  This bit can be set only if FS PHY interface is selected. Otherwise, this bit needs to be set to zero. If FS PHY interface is chosen and this bit is set, the PHY clock during Suspend must be switched from 48 MHz to 32 KHz.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): USB 1.1 Full-Speed Serial Transceiver not selected</li> <li>■ 0x1 (ENABLED): USB 1.1 Full-Speed Serial Transceiver Interface selected</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
2	NZstsOUTHShk	R/W	<p>Non-Zero-Length Status OUT Handshake (NZstsOUTHShk)  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> <ul style="list-style-type: none"> <li>■ 1'b1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.</li> <li>■ 1'b0: Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the Device Endpoint Control register.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (SENDOUT):  Send the received OUT packet to the application (zero-length or non-zero length) and send a handshake based on NAK and STALL bits for the endpoint in the Devce Endpoint Control Register</li> <li>■ 0x1 (SENDSTALL):  Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-52 Fields for Register: DCFG (Continued)**

Bits	Name	Memory Access	Description
1:0	DevSpd	R/W	<p>Device Speed (DevSpd)            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 connect sequence is completed, and is based on the speed of the USB host to which the core is connected.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (USBHS20): High speed USB 2.0 PHY clock is 30 MHz or 60 MHz</li> <li>■ 0x1 (USBFS20): Full speed USB 2.0 PHY clock is 30 MHz or 60 MHz</li> <li>■ 0x2 (USBLS116): Low speed USB 1.1 transceiver clock is 6 MHz</li> <li>■ 0x3 (USBFS1148): Full speed USB 1.1 transceiver clock is 48 MHz</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

## 7.1.48 DCTL

- **Name:** Device Control Register
  - **Description:** This register is used to control the characteristics of the Device controller.
  - **Size:** 32 bits
  - **Offset:** 0x804
  - **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==3 || OTG\_MODE == 4

RESERVED	31:20
ServInt	19
DeepSleepBESLReject	18
EnContOnBNA	17
NakOnBble	16
IgnFrmNum	15
GMC	14:13
Rsvd	12
PWROnPrgDone	11
CGOUTNak	10
SGOUTNak	9
CGNPInNak	8
SGNPInNak	7
TstCtl	6:4
GOUTNakSts	3
GNPINNakSts	2
SftDiscon	1
RmtWkUpSig	0

**Table 7-53 Fields for Register: DCTL**

Bits	Name	Memory Access	Description
31:20	RESERVED	R	RESERVED <b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead

**Table 7-53 Fields for Register: DCTL (Continued)**

Bits	Name	Memory Access	Description
19	ServInt	R/W	<p>Service Interval based scheduling for Isochronous IN Endpoints This field is used to enable service interval based scheduling feature for ISOC IN EPs.</p> <p><b>Note:</b> This bit is applicable only in device mode and when Scatter/Gather DMA mode is used. This feature should not be enabled along with DCTL.IgnrFrmNum.</p> <p><b>Scatter/Gather DMA Mode (GAHBCFG.DMAEn=1,DCFG.DescDMA=1):</b></p> <p>When this bit is enabled, the frame number field in the ISOC IN descriptor structure is interpreted as the last frame of the service interval. In Scatter/Gather DMA mode, if this bit is enabled, the pending packets are flushed by the controller at the last frame of the service interval.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): The controller behavior depends on DCTL.IgnrFrmNum field.</li> <li>■ 0x1 (ENABLED): Scatter/Gather DMA Mode: The controller can transmit the packets in any frame of the service interval.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Volatile:</b> true</p>
18	DeepSleepBESLReject	R/W	<p>DeepSleepBESLReject</p> <ul style="list-style-type: none"> <li>■ 1: Deep Sleep BESL Reject feature is enabled</li> <li>■ 0: Deep Sleep BESL Reject feature is disabled</li> </ul> <p>Core rejects LPM request with HIRD value greater than HIRD threshold programmed. NYET response is sent for LPM tokens with HIRD value greater than HIRD threshold. By default, the Deep Sleep BESL Reject feature is disabled.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Deep Sleep BESL Reject feature is disabled</li> <li>■ 0x1 (ENABLED): Deep Sleep BESL Reject feature is enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> (OTG_MODE==0    OTG_MODE==1    OTG_MODE==2    OTG_MODE==3    OTG_MODE==4) &amp;&amp; OTG_ENABLE_LPM==1</p>

**Table 7-53 Fields for Register: DCTL (Continued)**

Bits	Name	Memory Access	Description
17	EnContOnBNA	R/W	<p>Enable Continue on BNA (EnContOnBNA)  This bit enables the core to continue on BNA for Bulk OUT endpoints. With this feature enabled, when a Bulk OUT or INTR OUT endpoint receives a BNA interrupt the core starts processing the descriptor that caused the BNA interrupt after the endpoint re-enables the endpoint.</p> <ul style="list-style-type: none"> <li>■ 1'b0: After receiving BNA interrupt, the core disables the endpoint. When the endpoint is re-enabled by the application, the core starts processing from the DOEPDMA descriptor.</li> <li>■ 1'b1: After receiving BNA interrupt, the core disables the endpoint. When the endpoint is re-enabled by the application, the core starts processing from the descriptor that received the BNA interrupt.</li> </ul> <p>This bit is valid only when OTG_EN_DESC_DMA == 1'b1. It is a one-time programmable after reset bit like any other DCTL register bits.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Core disables the endpoint after receiving BNA interrupt. When application re-enables the endpoint, core starts processing from the DOEPDMA descriptor</li> <li>■ 0x1 (ENABLED): Core disables the endpoint after receiving BNA interrupt. When application re-enables the endpoint, core starts processing from the descriptor that received the BNA interrupt.</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_DESC_DMA == 1  <b>Testable:</b> readOnly</p>
16	NakOnBble	R/W	<p>NAK on Babble Error (NakOnBble)  Set NAK automatically on babble (NakOnBble). The core sets NAK automatically for the endpoint on which babble is received.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disable NAK on Babble Error</li> <li>■ 0x1 (ENABLED): NAK on Babble Error</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-53 Fields for Register: DCTL (Continued)**

Bits	Name	Memory Access	Description
15	IgnrFrmNum	R/W	<p>Ignore Frame Number Feature for Isochronous Endpoints (IgnrFrmNum)  This field is also used to control the Periodic Transfer Interrupt (PTI) feature.</p> <p><b>Note:</b> Do not program IgnrFrmNum bit to 1'b1 when the core is operating in threshold mode.</p> <p><b>Slave Mode (GAHBCFG.DMAEn=0):</b>  This bit is not valid in Slave mode and should not be programmed to 1.</p> <p><b>Scatter/Gather DMA Mode (GAHBCFG.DMAEn=1,DCFG.DescDMA=1):</b>  <b>Note:</b> When Scatter/Gather DMA mode is enabled this feature is not applicable to High Speed, High bandwidth transfers.  When this bit is enabled, there must be only one packet per descriptor.</p> <ul style="list-style-type: none"> <li>■ 0: The core transmits the packets only in the frame number in which they are intended to be transmitted.</li> <li>■ 1: The core ignores the frame number, sending packets immediately as the packets are ready.</li> </ul> <p>In Scatter/Gather DMA mode, if this bit is enabled, the packets are not flushed when a ISOC IN token is received for an elapsed frame.</p> <p><b>Non-Scatter/Gather DMA Mode, that is, Buffer DMA Mode (GAHBCFG.DMAEn=1,DCFG.DescDMA=0):</b>  When Scatter/Gather DMA mode is disabled, this field is used by the application to enable Periodic Transfer Interrupt (PTI) Mode.  The application can program Periodic Endpoint transfers for multiple (micro)Frames.</p> <ul style="list-style-type: none"> <li>■ 0: Periodic Transfer Interrupt feature is disabled, application needs to program transfers for periodic endpoints every (micro)Frame</li> <li>■ 1: Periodic Transfer Interrupt feature is enabled, application can program transfers for multiple (micro)Frames for periodic endpoints.</li> </ul>

**Table 7-53 Fields for Register: DCTL (Continued)**

Bits	Name	Memory Access	Description
			<p>In the PTI mode, the application will receive Transfer Complete Interrupt after transfers for multiple (micro)Frames are completed.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Scatter/Gather DMA Mode: The core transmits the packets only in the frame number in which they are intended to be transmitted. Non-Scatter/Gather DMA Mode: Periodic Transfer Interrupt feature is disabled.</li> <li>■ 0x1 (ENABLED): Scatter/Gather DMA Mode: The core ignores the frame number, sending packets immediately as the packets are ready. Non-Scatter/Gather DMA Mode: Periodic Transfer Interrupt feature is enabled.</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
14:13	GMC	R/W	<p>Global Multi Count (GMC)  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 non-periodic endpoints.</p> <ul style="list-style-type: none"> <li>■ 2'b00: Invalid.</li> <li>■ 2'b01: 1 packet.</li> <li>■ 2'b10: 2 packets.</li> <li>■ 2'b11: 3 packets.</li> </ul> <p>The value of this field automatically changes to 2'h1 when DCFG.DescDMA is set to 1. When Scatter/Gather DMA mode is disabled, this field is reserved, and reads 2'b00.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTVALID): Invalid</li> <li>■ 0x1 (ONEPACKET): 1 packet</li> <li>■ 0x2 (TWOPACKET): 2 packets</li> <li>■ 0x3 (THREEPACKET): 3 packets</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_DESC_DMA == 1  <b>Testable:</b> writeAsRead</p>
12			<b>Reserved Field:</b> Yes

**Table 7-53 Fields for Register: DCTL (Continued)**

Bits	Name	Memory Access	Description
11	PWROnPrgDone	R/W	<p>Power-On Programming Done (PWROnPrgDone)  The application uses this bit to indicate that register programming is completed after a wake-up from Power Down mode.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOTDONE): Power-On Programming not done</li> <li>■ 0x1 (DONE): Power-On Programming Done</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
10	CGOUTNak	W	<p>Clear Global OUT NAK (CGOUTNak)  A write to this field clears the Global OUT NAK.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disable Clear Global OUT NAK</li> <li>■ 0x1 (ENABLED): Clear Global OUT NAK</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
9	SGOUTNak	W	<p>Set Global OUT NAK (SGOUTNak)  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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disable Global OUT NAK</li> <li>■ 0x1 (ENABLED): Set Global OUT NAK</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
8	CGNPInNak	W	<p>Clear Global Non-periodic IN NAK (CGNPInNak)  A write to this field clears the Global Non-periodic IN NAK.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Disable Global Non-periodic IN NAK</li> <li>■ 0x1 (ENABLE): Clear Global Non-periodic IN NAK</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-53 Fields for Register: DCTL (Continued)**

Bits	Name	Memory Access	Description
7	SGNPIInNak	W	<p>Set Global Non-periodic IN NAK (SGNPIInNak)  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</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Disable Global Non-periodic IN NAK</li> <li>■ 0x1 (ENABLE): Set Global Non-periodic IN NAK</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
6:4	TstCtl	R/W	<p>Test Control (TstCtl)</p> <ul style="list-style-type: none"> <li>■ 3'b000: Test mode disabled</li> <li>■ 3'b001: Test_J mode</li> <li>■ 3'b010: Test_K mode</li> <li>■ 3'b011: Test_SE0_NAK mode</li> <li>■ 3'b100: Test_Packet mode</li> <li>■ 3'b101: Test_Force_Enable</li> <li>■ Others: Reserved</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Test mode disabled</li> <li>■ 0x1 (TESTJ): Test_J mode</li> <li>■ 0x2 (TESTK): Test_K mode</li> <li>■ 0x3 (TESTSN): Test_SE0_NAK mode</li> <li>■ 0x4 (TESTPM): Test_Packet mode</li> <li>■ 0x5 (TESTFE): Test_force_Enable</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-53 Fields for Register: DCTL (Continued)**

Bits	Name	Memory Access	Description
3	GOUTNaksts	R	<p>Global OUT NAK Status (GOUTNaksts)</p> <ul style="list-style-type: none"> <li>■ 1'b0: A handshake is sent based on the FIFO Status and the NAK and STALL bit settings.</li> <li>■ 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.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): A handshake is sent based on the FIFO Status and the NAK and STALL bit settings.</li> <li>■ 0x1 (ACTIVE): 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.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> writeAsRead</p>
2	GNPINNaksts	R	<p>Global Non-periodic IN NAK Status (GNPINNaksts)</p> <ul style="list-style-type: none"> <li>■ 1'b0: A handshake is sent out based on the data availability in the transmit FIFO.</li> <li>■ 1'b1: A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): A handshake is sent out based on the data availability in the transmit FIFO</li> <li>■ 0x1 (ACTIVE): A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> writeAsRead</p>

**Table 7-53 Fields for Register: DCTL (Continued)**

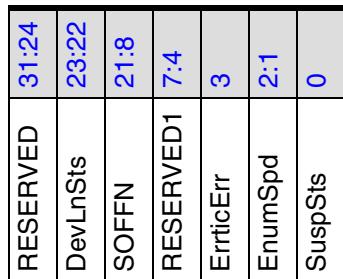
Bits	Name	Memory Access	Description
1	SftDiscon	R/W	<p>Soft Disconnect (SftDiscon)  The application uses this bit to signal the controller to do a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.</p> <ul style="list-style-type: none"> <li>■ 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.</li> <li>■ 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.</li> </ul> <p>The following is the minimum duration under various conditions for which this bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application adds some extra delay to the specified minimum duration.</p> <p>For high speed, if the device state is,</p> <ul style="list-style-type: none"> <li>■ Suspended, the minimum duration is 1ms + 2.5us</li> <li>■ Idle, the minimum duration is 3ms + 2.5us</li> <li>■ Not Idle or Suspended (performing transactions), the minimum duration 125 us</li> </ul> <p>For full speed/low speed, if the device state is,</p> <ul style="list-style-type: none"> <li>■ Suspended, the minimum duration is 1ms + 2.5us</li> <li>■ Idle, the minimum duration is 2.5us</li> <li>■ Not Idle or Suspended (performing transactions), the minimum duration 125 us</li> </ul> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>■ This bit can be also used for ULPI/FS Serial interfaces.</li> <li>■ This bit is not impacted by a soft reset.</li> </ul> <p><b>Values:</b></p>

**Table 7-53 Fields for Register: DCTL (Continued)**

Bits	Name	Memory Access	Description
			<ul style="list-style-type: none"> <li>■ 0x0 (NODISCONNECT): The core drives the phy_opmode_o signal on the UTMI+ to 2'b00, which generates a device connect event to the USB host</li> <li>■ 0x1 (DISCONNECT): The core drives the phy_opmode_o signal on the UTMI+ to 2'b01, which generates a device disconnect event to the USB host</li> </ul> <p><b>Value After Reset:</b> 0x1  <b>Exists:</b> Always</p>
0	RmtWkUpSig	R/W	<p>Remote Wakeup Signaling (RmtWkUpSig)  When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must Set this bit to instruct the core to exit the Suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1-15 ms after setting it.</p> <p>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 microseconds (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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLEDRMWKUP): Core does not send Remote Wakeup Signaling</li> <li>■ 0x1 (ENABLERMWKUP): Core sends Remote Wakeup Signaling</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

### 7.1.49 DSTS

- **Name:** Device Status Register
- **Description:** This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from Device All Interrupts (DAINT) register.
- **Size:** 32 bits
- **Offset:** 0x808
- **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==3 || OTG\_MODE == 4



**Table 7-54 Fields for Register: DSTS**

Bits	Name	Memory Access	Description
31:24	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
23:22	DevLnSts	R	<p>Device Line Status (DevLnSts)</p> <p>Indicates the current logic level USB data lines</p> <ul style="list-style-type: none"> <li>■ DevLnSts[1]: Logic level of D+</li> <li>■ DevLnSts[0]: Logic level of D-</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>

**Table 7-54 Fields for Register: DSTS (Continued)**

Bits	Name	Memory Access	Description
21:8	SOFFN	R	<p>Frame or Microframe Number of the Received SOF (SOFFN)  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> <p>Note: This register may return a non-zero value if read immediately after power-on reset. In case the register bit reads non-zero immediately after power-on reset, it does not indicate that SOF has been received from the host. The read value of this interrupt is valid only after a valid connection between host and device is established.</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always  <b>Volatile:</b> true</p>
7:4	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> 0  <b>Testable:</b> writeAsRead  <b>Write Constraint:</b> writeAsRead</p>
3	ErrticErr	R	<p>Erratic Error (ErrticErr)  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 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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Erratic Error</li> <li>■ 0x1 (ACTIVE): Erratic Error</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-54 Fields for Register: DSTS (Continued)**

Bits	Name	Memory Access	Description
2:1	EnumSpd	R	<p>Enumerated Speed (EnumSpd)      Indicates the speed at which the controller has come up after speed detection through a connect or reset sequence.</p> <ul style="list-style-type: none"> <li>■ 2'b00: High speed (PHY clock is running at 30 or 60 MHz)</li> <li>■ 2'b01: Full speed (PHY clock is running at 30 or 60 MHz)</li> <li>■ 2'b10: Low speed (PHY clock is running at 6 MHz)</li> <li>■ 2'b11: Full speed (PHY clock is running at 48 MHz)</li> </ul> <p>Low speed is not supported for devices using a UTMI+ PHY.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (HS3060): High speed (PHY clock is running at 30 or 60 MHz)</li> <li>■ 0x1 (FS3060): Full speed (PHY clock is running at 30 or 60 MHz)</li> <li>■ 0x2 (LS6): Low speed (PHY clock is running at 6 MHz)</li> <li>■ 0x3 (FS48): Full speed (PHY clock is running at 48 MHz)</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>
0	SuspSts	R	<p>Suspend Status (SuspSts)      In Device mode, this bit is set as long as a Suspend condition is detected on the USB. The core enters the Suspend state when there is no activity on the phy_line_state_i signal for an extended period of time. The core comes out of the suspend under the following conditions :</p> <ul style="list-style-type: none"> <li>■ If there is any activity on the phy_line_state_i signal, or</li> <li>■ If the application writes to the Remote Wakeup Signaling bit in the Device Control register (DCTL.RmtWkUpSig).</li> </ul> <p>When the core comes out of the suspend, this bit is set to 1'b0.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No suspend state</li> <li>■ 0x1 (ACTIVE): Suspend state</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>

### 7.1.50 DIEPMSK

- **Name:** Device IN Endpoint Common Interrupt Mask Register
- **Description:** This register works with each of the Device IN Endpoint Interrupt (DIEPINTn) registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the DIEPINTn register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.
- **Size:** 32 bits
- **Offset:** 0x810
- **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==3 || OTG\_MODE == 4

31:14	RESERVED	NAKMsk	Rsvd	12:10	BNAInIntrMsk	9	8	7	6	5	4	3	2	1	0
-------	----------	--------	------	-------	--------------	---	---	---	---	---	---	---	---	---	---

Table 7-55 Fields for Register: DIEPMSK

Bits	Name	Memory Access	Description
31:14	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
13	NAKMsk	R/W	<p>NAK interrupt Mask (NAKMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask NAK Interrupt</li> <li>■ 0x1 (NOMASK): No Mask NAK Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
12:10			<b>Reserved Field:</b> Yes

**Table 7-55 Fields for Register: DIEPMSK (Continued)**

Bits	Name	Memory Access	Description
9	BNAInIntrMsk	R/W	<p>BNA interrupt Mask (BNAInIntrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask BNA Interrupt</li> <li>■ 0x1 (NOMASK): No BNA Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA ==1</p>
8	TxfifoUndrnMsk	R/W	<p>Fifo Underrun Mask (TxfifoUndrnMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Fifo Underrun Interrupt</li> <li>■ 0x1 (NOMASK): No Fifo Underrun Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
7	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
6	INEPNakEffMsk	R/W	<p>IN Endpoint NAK Effective Mask (INEPNakEffMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint NAK Effective Interrupt</li> <li>■ 0x1 (NOMASK): No IN Endpoint NAK Effective Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
5	INTknEPMisMsk	R/W	<p>IN Token received with EP Mismatch Mask (INTknEPMisMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Token received with EP Mismatch Interrupt</li> <li>■ 0x1 (NOMASK): No Mask IN Token received with EP Mismatch Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-55 Fields for Register: DIEPMSK (Continued)**

Bits	Name	Memory Access	Description
4	INTknTxFEmpMsk	R/W	<p>IN Token Received When TxFIFO Empty Mask (INTknTxFEmpMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Token Received When TxFIFO Empty Interrupt</li> <li>■ 0x1 (NOMASK): No IN Token Received When TxFIFO Empty Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
3	TimeOUTMsk	R/W	<p>Timeout Condition Mask (TimeOUTMsk) (Non-isochronous endpoints)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Timeout Condition Interrupt</li> <li>■ 0x1 (NOMASK): No Timeout Condition Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
2	AHBErrMsk	R/W	<p>AHB Error Mask (AHBErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask AHB Error Interrupt</li> <li>■ 0x1 (NOMASK): No AHB Error Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE == 2</p>
1	EPDisbldMsk	R/W	<p>Endpoint Disabled Interrupt Mask (EPDisbldMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Endpoint Disabled Interrupt</li> <li>■ 0x1 (NOMASK): No Endpoint Disabled Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
0	XferComplMsk	R/W	<p>Transfer Completed Interrupt Mask (XferComplMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Transfer Completed Interrupt</li> <li>■ 0x1 (NOMASK): No Transfer Completed Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.51 DOEPMSK

- **Name:** Device OUT Endpoint Common Interrupt Mask Register
- **Description:** This register works with each of the Device OUT Endpoint Interrupt (DOEPINTn) registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the DOEPINTn register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.
- **Size:** 32 bits
- **Offset:** 0x814
- **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==3 || OTG\_MODE == 4

31:15	RESERVED	14	NYETMsk	NAKMsk	13	BbleErrMsk	12	Rsvd	11:10	BnaOutIntnMsk	9	OutPktErrMsk	8	RESERVED1	7	Back2BackSETup	6	StsPhaseRcvdMsk	5	OUTTknEPdisMsk	4	SetUPMsk	3	AHBERrMsk	2	EPDisbldMsk	1	XferComplMsk	0
-------	----------	----	---------	--------	----	------------	----	------	-------	---------------	---	--------------	---	-----------	---	----------------	---	-----------------	---	----------------	---	----------	---	-----------	---	-------------	---	--------------	---

Table 7-56 Fields for Register: DOEPMSK

Bits	Name	Memory Access	Description
31:15	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
14	NYETMsk	R/W	<p>NYET interrupt Mask (NYETMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask NYET Interrupt</li> <li>■ 0x1 (NOMASK): No NYET Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-56 Fields for Register: DOEPMSK (Continued)**

Bits	Name	Memory Access	Description
13	NAKMsk	R/W	<p>NAK interrupt Mask (NAKMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask NAK Interrupt</li> <li>■ 0x1 (NOMASK): No NAK Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
12	BbleErrMsk	R/W	<p>Babble Error interrupt Mask (BbleErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Babble Error Interrupt</li> <li>■ 0x1 (NOMASK): No Babble Error Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
11:10			<b>Reserved Field:</b> Yes
9	BnaOutIntrMsk	R/W	<p>BNA interrupt Mask (BnaOutIntrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask BNA Interrupt</li> <li>■ 0x1 (NOMASK): No BNA Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA ==1</p>
8	OutPktErrMsk	R/W	<p>OUT Packet Error Mask (OutPktErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Packet Error Interrupt</li> <li>■ 0x1 (NOMASK): No OUT Packet Error Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
7	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-56 Fields for Register: DOEPMSK (Continued)**

Bits	Name	Memory Access	Description
6	Back2BackSETUp	R/W	<p>Back-to-Back SETUP Packets Received Mask (Back2BackSETUp) Applies to control OUT endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Back-to-Back SETUP Packets Received Interrupt</li> <li>■ 0x1 (NOMASK): No Back-to-Back SETUP Packets Received Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_ARCHITECTURE !=0</p>
5	StsPhseRcvdMsk	R/W	<p>Status Phase Received Mask (StsPhseRcvdMsk) Applies to control OUT endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Status Phase Received Mask</li> <li>■ 0x1 (NOMASK): No Status Phase Received Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
4	OUTTknEPdisMsk	R/W	<p>OUT Token Received when Endpoint Disabled Mask (OUTTknEPdisMsk) Applies to control OUT endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Token Received when Endpoint Disabled Interrupt</li> <li>■ 0x1 (NOMASK): No OUT Token Received when Endpoint Disabled Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
3	SetUPMsk	R/W	<p>SETUP Phase Done Mask (SetUPMsk) Applies to control endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask SETUP Phase Done Interrupt</li> <li>■ 0x1 (NOMASK): No SETUP Phase Done Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

**Table 7-56 Fields for Register: DOEPMSK (Continued)**

Bits	Name	Memory Access	Description
2	AHBErrMsk	R/W	<p>AHB Error (AHBErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask AHB Error Interrupt</li> <li>■ 0x1 (NOMASK): No AHB Error Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE == 2</p>
1	EPDisbldMsk	R/W	<p>Endpoint Disabled Interrupt Mask (EPDisbldMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Endpoint Disabled Interrupt</li> <li>■ 0x1 (NOMASK): No Endpoint Disabled Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
0	XferComplMsk	R/W	<p>Transfer Completed Interrupt Mask (XferComplMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Transfer Completed Interrupt</li> <li>■ 0x1 (NOMASK): No Transfer Completed Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.52 DAINT

- **Name:** Device All Endpoints Interrupt Register
- **Description:** When a significant event occurs on an endpoint, a Device All Endpoints Interrupt register interrupts the application using the Device OUT Endpoints Interrupt bit or Device IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPIInt or GINTSTS.IEPInt, respectively). This is shown in Figure 5-2. There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Device Endpoint-n Interrupt register (DIEPINTn/DOEPINTn).
- **Size:** 32 bits
- **Offset:** 0x818
- **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==3 || OTG\_MODE == 4

OutEPInt15	31
OutEPInt14	30
OutEPInt13	29
OutEPInt12	28
OutEPInt11	27
OutEPInt10	26
OutEPInt9	25
OutEPInt8	24
OutEPInt7	23
OutEPInt6	22
OutEPInt5	21
OutEPInt4	20
OutEPInt3	19
OutEPInt2	18
OutEPInt1	17
OutEPInt0	16
InEPInt15	15
InEPInt14	14
InEPInt13	13
InEPInt12	12
InEPInt11	11
InEPInt10	10
InEPInt9	9
InEPInt8	8
InEPInt7	7
InEPInt6	6
InEPInt5	5
InEPInt4	4
InEPInt3	3
InEPInt2	2
InEPInt1	1
InEPInt0	0

Table 7-57 Fields for Register: DAINT

Bits	Name	Memory Access	Description
31	OutEPInt15	R	<p>OUT Endpoint 15 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_15 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 15</p>
30	OutEPInt14	R	<p>OUT Endpoint 14 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_14 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 14</p>

**Table 7-57 Fields for Register: DAINT (Continued)**

Bits	Name	Memory Access	Description
29	OutEPInt13	R	<p>OUT Endpoint 13 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_13 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 13</p>
28	OutEPInt12	R	<p>OUT Endpoint 12 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_12 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 12</p>
27	OutEPInt11	R	<p>OUT Endpoint 11 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_11 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 11</p>
26	OutEPInt10	R	<p>OUT Endpoint 10 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_10 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 10</p>
25	OutEPInt9	R	<p>OUT Endpoint 9 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_9 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 9</p>
24	OutEPInt8	R	<p>OUT Endpoint 8 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_8 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 8</p>

**Table 7-57 Fields for Register: DAINT (Continued)**

Bits	Name	Memory Access	Description
23	OutEPInt7	R	<p>OUT Endpoint 7 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_7 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 7</p>
22	OutEPInt6	R	<p>OUT Endpoint 6 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_6 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 6</p>
21	OutEPInt5	R	<p>OUT Endpoint 5 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_5 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 5</p>
20	OutEPInt4	R	<p>OUT Endpoint 4 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_4 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 4</p>
19	OutEPInt3	R	<p>OUT Endpoint 3 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_3 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 3</p>
18	OutEPInt2	R	<p>OUT Endpoint 2 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_2 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 2</p>

**Table 7-57 Fields for Register: DAINT (Continued)**

Bits	Name	Memory Access	Description
17	OutEPInt1	R	<p>OUT Endpoint 1 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the OUT EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_1 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 1</p>
16	OutEPInt0	R	<p>OUT Endpoint 0 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for OUT EP0</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
15	InEPInt15	R	<p>IN Endpoint 15 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_15 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 15</p>
14	InEPInt14	R	<p>IN Endpoint 14 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_14 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 14</p>
13	InEPInt13	R	<p>IN Endpoint 13 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_13 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 13</p>
12	InEPInt12	R	<p>IN Endpoint 12 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_12 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 12</p>

**Table 7-57 Fields for Register: DAINT (Continued)**

Bits	Name	Memory Access	Description
11	InEplnt11	R	<p>IN Endpoint 11 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_11 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 11</p>
10	InEplnt10	R	<p>IN Endpoint 10 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_10 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 10</p>
9	InEplnt9	R	<p>IN Endpoint 9 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_9 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 9</p>
8	InEplnt8	R	<p>IN Endpoint 8 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_8 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 8</p>
7	InEplnt7	R	<p>IN Endpoint 7 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_7 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 7</p>
6	InEplnt6	R	<p>IN Endpoint 6 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_6 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 6</p>

**Table 7-57 Fields for Register: DAINT (Continued)**

Bits	Name	Memory Access	Description
5	InEplnt5	R	<p>IN Endpoint 5 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_5 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 5</p>
4	InEplnt4	R	<p>IN Endpoint 4 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_4 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 4</p>
3	InEplnt3	R	<p>IN Endpoint 3 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_3 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 3</p>
2	InEplnt2	R	<p>IN Endpoint 2 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_2 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 2</p>
1	InEplnt1	R	<p>IN Endpoint 1 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for the IN EP</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_1 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 1</p>
0	InEplnt0	R	<p>IN Endpoint 0 Interrupt Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Interrupt</li> <li>■ 0x1 (ACTIVE): Interrupt is active for IN EP0</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.53 DAINTMSK

- **Name:** Device All Endpoints Interrupt Mask Register
- **Description:** The Device Endpoint Interrupt Mask register works with the Device Endpoint Interrupt register to interrupt the application when an event occurs on a device endpoint. However, the Device All Endpoints Interrupt (DAINT) register bit corresponding to that interrupt is still set.
- **Size:** 32 bits
- **Offset:** 0x81c
- **Exists:** OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==3 || OTG\_MODE == 4

OutEPMsk15	31	OutEPMsk14	30	OutEPMsk13	29	OutEPMsk12	28	OutEPMsk11	27	OutEPMsk10	26	OutEPMsk9	25	OutEPMsk8	24	OutEPMsk7	23	OutEPMsk6	22	OutEPMsk5	21	OutEPMsk4	20	OutEPMsk3	19	OutEPMsk2	18	OutEPMsk1	17	OutEPMsk0	16	InEpMsk15	15	InEpMsk14	14	InEpMsk13	13	InEpMsk12	12	InEpMsk11	11	InEpMsk10	10	InEpMsk9	9	InEpMsk8	8	InEpMsk7	7	InEpMsk6	6	InEpMsk5	5	InEpMsk4	4	InEpMsk3	3	InEpMsk2	2	InEpMsk1	1	InEpMsk0	0
------------	----	------------	----	------------	----	------------	----	------------	----	------------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	-----------	----	----------	---	----------	---	----------	---	----------	---	----------	---	----------	---	----------	---	----------	---	----------	---	----------	---

Table 7-58 Fields for Register: DAINTMSK

Bits	Name	Memory Access	Description
31	OutEPMsk15	R/W	<p>OUT Endpoint 15 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_15 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 15</p>
30	OutEPMsk14	R/W	<p>OUT Endpoint 14 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_14 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 14</p>

**Table 7-58 Fields for Register: DAINTMSK (Continued)**

Bits	Name	Memory Access	Description
29	OutEPMsk13	R/W	<p>OUT Endpoint 13 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_13 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 13</p>
28	OutEPMsk12	R/W	<p>OUT Endpoint 12 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_12 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 12</p>
27	OutEPMsk11	R/W	<p>OUT Endpoint 11 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_11 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 11</p>
26	OutEPMsk10	R/W	<p>OUT Endpoint 10 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_10 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 10</p>
25	OutEPMsk9	R/W	<p>OUT Endpoint 9 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_9 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 9</p>
24	OutEPMsk8	R/W	<p>OUT Endpoint 8 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_8 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 8</p>

**Table 7-58 Fields for Register: DAINTMSK (Continued)**

Bits	Name	Memory Access	Description
23	OutEPMsk7	R/W	<p>OUT Endpoint 7 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_7 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 7</p>
22	OutEPMsk6	R/W	<p>OUT Endpoint 6 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_6 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 6</p>
21	OutEPMsk5	R/W	<p>OUT Endpoint 5 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_5 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 5</p>
20	OutEPMsk4	R/W	<p>OUT Endpoint 4 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_4 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 4</p>
19	OutEPMsk3	R/W	<p>OUT Endpoint 3 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_3 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 3</p>
18	OutEPMsk2	R/W	<p>OUT Endpoint 2 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_2 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 2</p>

**Table 7-58 Fields for Register: DAINTMSK (Continued)**

Bits	Name	Memory Access	Description
17	OutEPMsk1	R/W	<p>OUT Endpoint 1 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_1 !=1 &amp;&amp; OTG_NUM_EPS &gt;= 1</p>
16	OutEPMsk0	R/W	<p>OUT Endpoint 0 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Endpoint 0 Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
15	InEpMsk15	R/W	<p>IN Endpoint 15 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_15 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 15</p>
14	InEpMsk14	R/W	<p>IN Endpoint 14 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_14 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 14</p>
13	InEpMsk13	R/W	<p>IN Endpoint 13 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_13 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 13</p>
12	InEpMsk12	R/W	<p>IN Endpoint 12 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_12 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 12</p>

**Table 7-58 Fields for Register: DAINTMSK (Continued)**

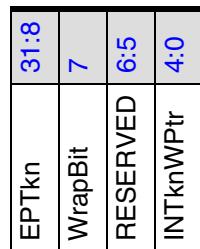
Bits	Name	Memory Access	Description
11	InEpMsk11	R/W	<p>IN Endpoint 11 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_11 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 11</p>
10	InEpMsk10	R/W	<p>IN Endpoint 10 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_10 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 10</p>
9	InEpMsk9	R/W	<p>IN Endpoint 9 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_9 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 9</p>
8	InEpMsk8	R/W	<p>IN Endpoint 8 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_8 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 8</p>
7	InEpMsk7	R/W	<p>IN Endpoint 7 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_7 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 7</p>
6	InEpMsk6	R/W	<p>IN Endpoint 6 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_6 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 6</p>

**Table 7-58 Fields for Register: DAINTMSK (Continued)**

Bits	Name	Memory Access	Description
5	InEpMsk5	R/W	<p>IN Endpoint 5 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_5 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 5</p>
4	InEpMsk4	R/W	<p>IN Endpoint 4 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_4 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 4</p>
3	InEpMsk3	R/W	<p>IN Endpoint 3 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_3 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 3</p>
2	InEpMsk2	R/W	<p>IN Endpoint 2 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_2 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 2</p>
1	InEpMsk1	R/W	<p>IN Endpoint 1 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EP_DIR_1 !=2 &amp;&amp; OTG_NUM_EPS &gt;= 1</p>
0	InEpMsk0	R/W	<p>IN Endpoint 0 Interrupt mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint 0 Interrupt</li> <li>■ 0x1 (NOMASK): No Interrupt mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.54 DTKNQR1

- **Name:** Device IN Token Sequence Learning Queue Read Register 1
- **Description:** This register is valid only in non-periodic Shared FIFO operation (OTG\_ENDED\_TX\_FIFO = 0). The depth of the IN Token Sequence Learning Queue is specified for Device Mode IN Token Sequence Learning Queue Depth during coreConsultant configuration (parameter OTG\_TOKEN\_QUEUE\_DEPTH). The queue is 4 bits wide to store the endpoint number. A read from this register returns the first 5 endpoint entries of the IN Token Sequence Learning Queue. When the queue is full, the new token is pushed into the queue and oldest token is discarded.
- **Size:** 32 bits
- **Offset:** 0x820
- **Exists:** (OTG\_MODE == 0 || OTG\_MODE == 2 || OTG\_MODE == 1 || OTG\_MODE == 3 || OTG\_MODE == 4) && OTG\_ENDED\_TX\_FIFO == 0



**Table 7-59 Fields for Register: DTKNQR1**

Bits	Name	Memory Access	Description
31:8	EPTkn	R	<p>Endpoint Token (EPTkn)</p> <p>Four bits per token represent the endpoint number of the token:</p> <ul style="list-style-type: none"> <li>■ Bits [31:28]: Endpoint number of Token 5</li> <li>■ Bits [27:24]: Endpoint number of Token 4, .....</li> <li>■ Bits [15:12]: Endpoint number of Token 1</li> <li>■ Bits [11:8]: Endpoint number of Token 0</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENDED_TX_FIFO == 0</p>
7	WrapBit	R	<p>Wrap Bit (WrapBit) This bit is Set when the write pointer wraps. It is cleared when the learning queue is cleared.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLE): Learning Queue is cleared</li> <li>■ 0x1 (ENABLE): Write Pointer wraps</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENDED_TX_FIFO == 0</p>

**Table 7-59 Fields for Register: DTKNQR1 (Continued)**

Bits	Name	Memory Access	Description
6:5	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
4:0	INTknWPtr	R	<p>IN Token Queue Write Pointer (INTknWPtr)</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DED_TX_FIFO == 0</p>

### 7.1.55 DTKNQR2

- **Name:** Device IN Token Sequence Learning Queue Read Register 2
- **Description:** This register is valid only in shared non-periodic Shared FIFO operation (OTG\_ENDED\_TX\_FIFO == 0). A read from this register returns the next 8 endpoint entries of the learning queue.
- **Size:** 32 bits
- **Offset:** 0x824
- **Exists:** (OTG\_MODE == 0 || OTG\_MODE == 2 || OTG\_MODE == 1 || OTG\_MODE == 3 || OTG\_MODE == 4) && OTG\_ENDED\_TX\_FIFO == 0

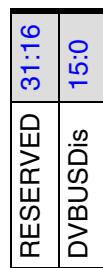


**Table 7-60 Fields for Register: DTKNQR2**

Bits	Name	Memory Access	Description
31:0	EPTkn	R	<p>Endpoint Token (EPTkn) Four bits per token represent the endpoint number of the token:</p> <ul style="list-style-type: none"> <li>■ Bits [31:28]: Endpoint number of Token 13</li> <li>■ Bits [27:24]: Endpoint number of Token 12 .....</li> <li>■ Bits [7:4]: Endpoint number of Token 7</li> <li>■ Bits [3:0]: Endpoint number of Token 6</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_ENDED_TX_FIFO == 0</p>

### 7.1.56 DVBUSDIS

- **Name:** Device VBUS Discharge Time Register
- **Description:** This register specifies the VBUS discharge time after VBUS pulsing during SRP.
- **Size:** 32 bits
- **Offset:** 0x828
- **Exists:** (OTG\_MODE == 0 || OTG\_MODE == 2 || OTG\_MODE == 1 || OTG\_MODE == 3 || OTG\_MODE == 4)

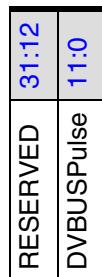


**Table 7-61 Fields for Register: DVBUSDIS**

Bits	Name	Memory Access	Description
31:16	RESERVED	R	<b>RESERVED</b> <b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead
15:0	DVBUSDis	R/W	Device VBUS Discharge Time (DVBUSDis) 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 30MHz (16-bit data width) or 60 MHz (8-bit data width). Depending on your VBUS load, this value can need adjustment. <b>Value After Reset:</b> 0x17d7 <b>Exists:</b> Always

### 7.1.57 DVBUSPULSE

- **Name:** Device VBUS Pulsing Time Register
- **Description:** This register contains the VBUS pulsing time during SRP.
- **Size:** 32 bits
- **Offset:** 0x82c
- **Exists:** (OTG\_MODE ==0 || OTG\_MODE ==2 || OTG\_MODE == 1 || OTG\_MODE ==3 || OTG\_MODE == 4)



**Table 7-62 Fields for Register: DVBUSPULSE**

Bits	Name	Memory Access	Description
31:12	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
11:0	DVBUSPulse	R/W	<p>Device VBUS Pulsing Time (DVBUSPulse)</p> <p>Specifies the VBUS pulsing time during SRP. This value equals (VBUS pulsing time in PHY clocks) / 1, 024</p> <p>The value you use depends whether the PHY is operating at 30MHz (16-bit data width) or 60 MHz (8-bit data width).</p> <p><b>Value After Reset:</b> 0x5b8</p> <p><b>Exists:</b> Always</p>

### 7.1.58 DTHRCTL

- **Name:** Device Threshold Control Register
- **Description:** This register contains the Receive and Transmit Threshold characteristics of the Device controller.
- **Size:** 32 bits
- **Offset:** 0x830
- **Exists:** (OTG\_MODE == 0 || OTG\_MODE == 2 || OTG\_MODE == 1 || OTG\_MODE == 3 || OTG\_MODE == 4) && OTG\_ENDED\_TX\_FIFO == 1

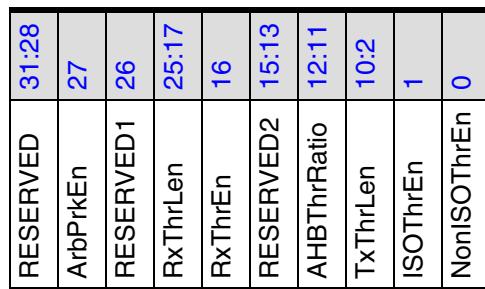


Table 7-63 Fields for Register: DTHRCTL

Bits	Name	Memory Access	Description
31:28	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
27	ArbPrkEn	OTG_AR CHTECT URE == 0 ? R:R/W	<p>Arbiter Parking Enable (ArbPrkEn)</p> <p>This bit controls internal DMA arbiter parking for IN endpoints. If 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, arbiter parking is enabled.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disable DMA arbiter parking</li> <li>■ 0x1 (ENABLED): Enable DMA arbiter parking for IN endpoints</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENDED_TX_FIFO == 1</p>

**Table 7-63 Fields for Register: DTHRCTL (Continued)**

Bits	Name	Memory Access	Description
26	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
25:17	RxThrLen	OTG_AR CHTECT URE == 0 ? R:R/W	<p>Receive Threshold Length (RxThrLen)</p> <p>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> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENDED_TX_FIFO == 1</p>
16	RxThrEn	OTG_AR CHTECT URE == 0 ? R:R/W	<p>Receive Threshold Enable (RxThrEn)</p> <p>When this bit is set, the core enables thresholding in the receive direction.</p> <p>Note: We recommends that you do not enable RxThrEn, because it may cause issues in the RxFIFO especially during error conditions such as RxError and Babble.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disable thresholding</li> <li>■ 0x1 (ENABLED): Enable thresholding in the receive direction</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENDED_TX_FIFO == 1</p>
15:13	RESERVED2	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-63 Fields for Register: DTHRCTL (Continued)**

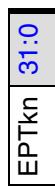
Bits	Name	Memory Access	Description
12:11	AHBThrRatio	OTG_AR CHTECT URE == 0 ? R:R/W	<p>AHB Threshold Ratio (AHBThrRatio)        These bits define the ratio between the AHB threshold and the MAC threshold for the transmit path only. The AHB threshold always remains less than or equal to the USB threshold, 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.</p> <ul style="list-style-type: none"> <li>■ 2'b00: AHB threshold = MAC threshold</li> <li>■ 2'b01: AHB threshold = MAC threshold / 2</li> <li>■ 2'b10: AHB threshold = MAC threshold / 4</li> <li>■ 2'b11: AHB threshold = MAC threshold / 8</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (THRESZERO): AHB threshold = MAC threshold</li> <li>■ 0x1 (THRESONE): AHB threshold = MAC threshold / 2</li> <li>■ 0x2 (THRESTWO): AHB threshold = MAC threshold / 4</li> <li>■ 0x3 (THRESTHREE): AHB threshold = MAC threshold / 8</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DED_TX_FIFO == 1</p>

**Table 7-63 Fields for Register: DTHRCTL (Continued)**

Bits	Name	Memory Access	Description
10:2	TxThrLen	OTG_AR CHTECT URE == 0 ? R:R/W	<p>Transmit Threshold Length (TxThrLen)  This field specifies Transmit thresholding size in DWORDS. This 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 transmit on the USB. The threshold length has to be at least eight DWORDS when the value of AHBThrRatio is 2'h00. In case the AHBThrRatio is non zero the application needs to ensure that the AHB Threshold value does not go below the recommended eight DWORD. This field controls both isochronous and non-isochronous IN endpoint thresholds. The recommended value for ThrLen is to be the same as the programmed AHB Burst Length (GAHBCFG.HBstLen).</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>■ When OTG_ARCHITECTURE=0, the reset value of this register field is 0.</li> <li>■ When OTG_ARCHITECTURE=2, the reset value of this register field is 8.</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_ENDED_TX_FIFO == 1</p>
1	ISOThrEn	OTG_AR CHTECT URE == 0 ? R:R/W	<p>ISO IN Endpoints Threshold Enable. (ISOThrEn)  When this bit is Set, the core enables thresholding for isochronous IN endpoints.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No thresholding</li> <li>■ 0x1 (ENABLED): Enables thresholding for isochronous IN endpoints</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_ENDED_TX_FIFO == 1</p>
0	NonISOThrEn	OTG_AR CHTECT URE == 0 ? R:R/W	<p>Non-ISO IN Endpoints Threshold Enable. (NonISOThrEn)  When this bit is Set, the core enables thresholding for Non Isochronous IN endpoints.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): No thresholding</li> <li>■ 0x1 (ENABLED): Enable thresholding for non-isochronous IN endpoints</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_ENDED_TX_FIFO == 1</p>

### 7.1.59 DTKNQR3

- **Name:** Device IN Token Sequence Learning Queue Read Register 3
- **Description:** This register is valid only in shared non-periodic Shared FIFO operation (OTG\_ENDED\_TX\_FIFO == 0). A read from this register returns the next 8 endpoint entries of the learning queue.
- **Size:** 32 bits
- **Offset:** 0x830
- **Exists:** (OTG\_MODE == 0 || OTG\_MODE == 2 || OTG\_MODE == 1 || OTG\_MODE == 3 || OTG\_MODE == 4) && OTG\_ENDED\_TX\_FIFO == 0



**Table 7-64 Fields for Register: DTKNQR3**

Bits	Name	Memory Access	Description
31:0	EPTkn	R	<p>Endpoint Token (EPTkn) Four bits per token represent the endpoint number of the token:</p> <ul style="list-style-type: none"> <li>■ Bits [31:28]: Endpoint number of Token 21</li> <li>■ Bits [27:24]: Endpoint number of Token 20 .....</li> <li>■ Bits [7:4]: Endpoint number of Token 15</li> <li>■ Bits [3:0]: Endpoint number of Token 14</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_ENDED_TX_FIFO == 0</p>

### 7.1.60 DIEPEMPMSK

- **Name:** Device IN Endpoint FIFO Empty Interrupt Mask Register
- **Description:** This register is valid only in Dedicated FIFO operation (OTG\_ENDED\_TX\_FIFO = 1). This register is used to control the IN endpoint FIFO empty interrupt generation (DIEPINTn.TxfEmp).
- **Size:** 32 bits
- **Offset:** 0x834
- **Exists:** (OTG\_MODE == 0 || OTG\_MODE == 2 || OTG\_MODE == 1 || OTG\_MODE == 3 || OTG\_MODE == 4) && OTG\_ENDED\_TX\_FIFO==1



**Table 7-65 Fields for Register: DIEPEMPMSK**

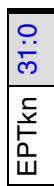
Bits	Name	Memory Access	Description
31:16	RESERVED	R	<b>RESERVED</b> <b>Value After Reset:</b> 0x0 <b>Exists:</b> 0 <b>Testable:</b> writeAsRead <b>Write Constraint:</b> writeAsRead

**Table 7-65 Fields for Register: DIEPEMPMSK (Continued)**

Bits	Name	Memory Access	Description
15:0	InEpTxfEmpMsk	R/W	<p>IN EP Tx FIFO Empty Interrupt Mask Bits (InEpTxfEmpMsk)      These bits acts as mask bits for DIEPINTn.TxFEmp interrupt, one bit per IN Endpoint:      Bit 0 for IN EP 0, bit 15 for IN EP 15</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (EP0_MASK): Mask IN EP0 Tx FIFO Empty Interrupt</li> <li>■ 0x2 (EP1_MASK): Mask IN EP1 Tx FIFO Empty Interrupt</li> <li>■ 0x4 (EP2_MASK): Mask IN EP2 Tx FIFO Empty Interrupt</li> <li>■ 0x8 (EP3_MASK): Mask IN EP3 Tx FIFO Empty Interrupt</li> <li>■ 0x10 (EP4_MASK): Mask IN EP4 Tx FIFO Empty Interrupt</li> <li>■ 0x20 (EP5_MASK): Mask IN EP5 Tx FIFO Empty Interrupt</li> <li>■ 0x40 (EP6_MASK): Mask IN EP6 Tx FIFO Empty Interrupt</li> <li>■ 0x80 (EP7_MASK): Mask IN EP7 Tx FIFO Empty Interrupt</li> <li>■ 0x100 (EP8_MASK): Mask IN EP8 Tx FIFO Empty Interrupt</li> <li>■ 0x200 (EP9_MASK): Mask IN EP9 Tx FIFO Empty Interrupt</li> <li>■ 0x400 (EP10_MASK): Mask IN EP10 Tx FIFO Empty Interrupt</li> <li>■ 0x800 (EP11_MASK): Mask IN EP11 Tx FIFO Empty Interrupt</li> <li>■ 0x1000 (EP12_MASK): Mask IN EP12 Tx FIFO Empty Interrupt</li> <li>■ 0x2000 (EP13_MASK): Mask IN EP13 Tx FIFO Empty Interrupt</li> <li>■ 0x4000 (EP14_MASK): Mask IN EP14 Tx FIFO Empty Interrupt</li> <li>■ 0x8000 (EP15_MASK): Mask IN EP15 Tx FIFO Empty Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENDED_TX_FIFO == 1</p>

### 7.1.61 DTKNQR4

- **Name:** Device IN Token Sequence Learning Queue Read Register 4
- **Description:** This register is valid only in shared non-periodic Shared FIFO operation (OTG\_ENDED\_TX\_FIFO == 0). A read from this register returns the next 8 endpoint entries of the learning queue.
- **Size:** 32 bits
- **Offset:** 0x834
- **Exists:** (OTG\_MODE == 0 || OTG\_MODE == 2 || OTG\_MODE == 1 || OTG\_MODE == 3 || OTG\_MODE == 4) && OTG\_ENDED\_TX\_FIFO == 0



**Table 7-66 Fields for Register: DTKNQR4**

Bits	Name	Memory Access	Description
31:0	EPTkn	R	<p>Endpoint Token (EPTkn) Four bits per token represent the endpoint number of the token:</p> <ul style="list-style-type: none"> <li>■ Bits [31:28]: Endpoint number of Token 29</li> <li>■ Bits [27:24]: Endpoint number of Token 28.....</li> <li>■ Bits [7:4]: Endpoint number of Token 23</li> <li>■ Bits [3:0]: Endpoint number of Token 22</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_ENDED_TX_FIFO == 0</p>

### 7.1.62 DEACHINT

- **Name:** Device Each Endpoints Interrupt Register
- **Description:** This register is available in device mode and only when parameter "OTG\_MULTI\_PROC\_INTRPT" on page 121=1. There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Device Endpoint-n Interrupt register (DIEPINTn/DOEPINTn). The interrupt is automatically cleared once the DOEPINTn/DIEPINTn interrupt is cleared by the application.
- **Size:** 32 bits
- **Offset:** 0x838
- **Exists:** OTG\_MULTI\_PROC\_INTRPT == 1

EchOutEPInt	31:16
EchInEPInt	15:0

**Table 7-67 Fields for Register: DEACHINT**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
31:16	EchOutEPInt	R	<p>Each OUT Endpoint Interrupt Bits (EchOutEPInt) One bit per OUT endpoint: Bit 16 for OUT endpoint 0, bit 31 for OUT endpoint 15</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP0_INT): OUT EP0 Interrupt</li> <li>■ 0x2 (OEP1_INT): OUT EP1 Interrupt</li> <li>■ 0x4 (OEP2_INT): OUT EP2 Interrupt</li> <li>■ 0x8 (OEP3_INT): OUT EP3 Interrupt</li> <li>■ 0x10 (OEP4_INT): OUT EP4 Interrupt</li> <li>■ 0x20 (OEP5_INT): OUT EP5 Interrupt</li> <li>■ 0x40 (OEP6_INT): OUT EP6 Interrupt</li> <li>■ 0x80 (OEP7_INT): OUT EP7 Interrupt</li> <li>■ 0x100 (OEP8_INT): OUT EP8 Interrupt</li> <li>■ 0x200 (OEP9_INT): OUT EP9 Interrupt</li> <li>■ 0x400 (OEP10_INT): OUT EP10 Interrupt</li> <li>■ 0x800 (OEP11_INT): OUT EP11 Interrupt</li> <li>■ 0x1000 (OEP12_INT): OUT EP12 Interrupt</li> <li>■ 0x2000 (OEP13_INT): OUT EP13 Interrupt</li> <li>■ 0x4000 (OEP14_INT): OUT EP14 Interrupt</li> <li>■ 0x8000 (OEP15_INT): OUT EP15 Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>

**Table 7-67 Fields for Register: DEACHINT (Continued)**

Bits	Name	Memory Access	Description
15:0	EchInEplnt	R	<p>Each IN Endpoint Interrupt Bits (EchInEplnt)  One bit per IN Endpoint:  Bit 0 for IN endpoint 0, bit 15 for endpoint 15</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP0_INT): IN EP0 Interrupt</li> <li>■ 0x2 (IEP1_INT): IN EP1 Interrupt</li> <li>■ 0x4 (IEP2_INT): IN EP2 Interrupt</li> <li>■ 0x8 (IEP3_INT): IN EP3 Interrupt</li> <li>■ 0x10 (IEP4_INT): IN EP4 Interrupt</li> <li>■ 0x20 (IEP5_INT): IN EP5 Interrupt</li> <li>■ 0x40 (IEP6_INT): IN EP6 Interrupt</li> <li>■ 0x80 (IEP7_INT): IN EP7 Interrupt</li> <li>■ 0x100 (IEP8_INT): IN EP8 Interrupt</li> <li>■ 0x200 (IEP9_INT): IN EP9 Interrupt</li> <li>■ 0x400 (IEP10_INT): IN EP10 Interrupt</li> <li>■ 0x800 (IEP11_INT): IN EP11 Interrupt</li> <li>■ 0x1000 (IEP12_INT): IN EP12 Interrupt</li> <li>■ 0x2000 (IEP13_INT): IN EP13 Interrupt</li> <li>■ 0x4000 (IEP14_INT): IN EP14 Interrupt</li> <li>■ 0x8000 (IEP15_INT): IN EP15 Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>

### 7.1.63 DEACHINTMSK

- **Name:** Device Each Endpoints Interrupt Mask Register
- **Description:** This register is available only when parameter OTG\_MULTI\_PROC\_INTRPT=1. The Device Each Endpoint Interrupt Mask register works with the Device Each Endpoint Interrupt register to interrupt the application when an event occurs on a device endpoint. However, the Device Each Endpoints Interrupt (DEACHINT) register bit corresponding to that interrupt remains set.
- **Size:** 32 bits
- **Offset:** 0x83c
- **Exists:** OTG\_MULTI\_PROC\_INTRPT==1

31	EchOutEpMsk15	30	EchOutEpMsk14	29	EchOutEpMsk13	28	EchOutEpMsk12	27	EchOutEpMsk11	26	EchOutEpMsk10	25	EchOutEpMsk9	24	EchOutEpMsk8	23	EchOutEpMsk7	22	EchOutEpMsk6	21	EchOutEpMsk5	20	EchOutEpMsk4	19	EchOutEpMsk3	18	EchOutEpMsk2	17	EchOutEpMsk1	16	EchOutEpMsk0	15	EchInEpMsk15	14	EchInEpMsk14	13	EchInEpMsk13	12	EchInEpMsk12	11	EchInEpMsk11	10	EchInEpMsk10	9	EchInEpMsk9	8	EchInEpMsk8	7	EchInEpMsk7	6	EchInEpMsk6	5	EchInEpMsk5	4	EchInEpMsk4	3	EchInEpMsk3	2	EchInEpMsk2	1	EchInEpMsk1	0	EchInEpMsk0
----	---------------	----	---------------	----	---------------	----	---------------	----	---------------	----	---------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	----	--------------	---	-------------	---	-------------	---	-------------	---	-------------	---	-------------	---	-------------	---	-------------	---	-------------	---	-------------	---	-------------

Table 7-68 Fields for Register: DEACHINTMSK

Bits	Name	Memory Access	Description
31	EchOutEpMsk15	* Varies	<p>OUT EP15 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP15_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS == 15</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 15</p>
30	EchOutEpMsk14	* Varies	<p>OUT EP14 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP14_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 14</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 14</p>

**Table 7-68 Fields for Register: DEACHINTMSK (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
29	EchOutEpMsk13	* Varies	<p>OUT EP13 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP13_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 13</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 13</p>
28	EchOutEpMsk12	* Varies	<p>OUT EP12 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP12_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 12</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 12</p>
27	EchOutEpMsk11	* Varies	<p>OUT EP11 Interrupt Mask Bits</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP11_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 11</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 11</p>
26	EchOutEpMsk10	* Varies	<p>OUT EP10 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP10_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 10</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 10</p>
25	EchOutEpMsk9	* Varies	<p>OUT EP9 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP9_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 9</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 9</p>
24	EchOutEpMsk8	* Varies	<p>OUT EP8 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP8_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 8</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 8</p>

**Table 7-68 Fields for Register: DEACHINTMSK (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
23	EchOutEpMsk7	* Varies	<p>OUT EP7 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP7_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 7</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 7</p>
22	EchOutEpMsk6	* Varies	<p>OUT EP6 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP6_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 6</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 6</p>
21	EchOutEpMsk5	* Varies	<p>OUT EP5 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP5_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 5</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 5</p>
20	EchOutEpMsk4	* Varies	<p>OUT EP4 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP4_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 4</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 4</p>
19	EchOutEpMsk3	* Varies	<p>OUT EP3 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP3_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 3</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 3</p>
18	EchOutEpMsk2	* Varies	<p>OUT EP2 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP2_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 2</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 2</p>

**Table 7-68 Fields for Register: DEACHINTMSK (Continued)**

Bits	Name	Memory Access	Description
17	EchOutEpMsk1	* Varies	<p>OUT EP1 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP1_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 1</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 1</p>
16	EchOutEpMsk0	R/W	<p>OUT EP0 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (OEP0_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
15	EchInEpMsk15	* Varies	<p>IN EP15 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP15_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS == 15</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 15</p>
14	EchInEpMsk14	* Varies	<p>IN EP14 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP14_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 14</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 14</p>
13	EchInEpMsk13	* Varies	<p>IN EP13 Interrupt Mask Bits</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP13_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 13</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 13</p>
12	EchInEpMsk12	* Varies	<p>IN EP12 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP12_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 12</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 12</p>

**Table 7-68 Fields for Register: DEACHINTMSK (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
11	EchInEpMsk11	* Varies	<p>IN EP11 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP11_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 11</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 11</p>
10	EchInEpMsk10	* Varies	<p>IN EP10 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP10_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 10</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 10</p>
9	EchInEpMsk9	* Varies	<p>IN EP9 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP9_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 9</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 9</p>
8	EchInEpMsk8	* Varies	<p>IN EP8 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP8_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 8</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 8</p>
7	EchInEpMsk7	* Varies	<p>IN EP7 Interrupt Mask Bits</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP7_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 7</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 7</p>
6	EchInEpMsk6	* Varies	<p>IN EP6 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP6_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 6</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 6</p>

**Table 7-68 Fields for Register: DEACHINTMSK (Continued)**

Bits	Name	Memory Access	Description
5	EchInEpMsk5	* Varies	<p>IN EP5 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP5_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 5</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 5</p>
4	EchInEpMsk4	* Varies	<p>IN EP4 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP4_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 4</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 4</p>
3	EchInEpMsk3	* Varies	<p>IN EP3 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP3_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 3</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 3</p>
2	EchInEpMsk2	* Varies	<p>IN EP2 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP2_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 2</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 2</p>
1	EchInEpMsk1	* Varies	<p>IN EP1 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP1_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_NUM_EPS &gt;= 1</p> <p><b>Memory Access:</b> =&lt;functionof&gt; 1</p>
0	EchInEpMsk0	R/W	<p>IN EP0 Interrupt Mask Bit</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (IEP0_INT_MASK)</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.64 DIEPEACHMSK0

- **Name:** Device Each IN Endpoint 0 Interrupt Register
- **Description:** This register is available in device mode and only when parameter "OTG\_MULTI\_PROC\_INTRPT" on page 121=1. These registers are endpoint-specific mask registers for (DIEPINTn). The IN endpoint interrupt for a specific status in the DIEPINTn register can be masked by writing 0 to the corresponding bit in this register. Status bits are masked by default.
  - Mask interrupt: 1'b0
  - Unmask interrupt: 1'b1
- **Size:** 32 bits
- **Offset:** 0x840
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 0) && OTG\_MULTI\_PROC\_INTRPT == 1

31:14														
Rsvd	NAKMsk	Rsvd	12:10	9	8	7	RESERVED	INEPNakEffMsk	6	INTknEPMisMsk	5	INTknTXFEmpMsk	4	TimeOUTMsk
				TxfifoUndrmSk										AHBErrMsk
														EPDisblIdMsk
														XferComplMsk
														0

Table 7-69 Fields for Register: DIEPEACHMSK0

Bits	Name	Memory Access	Description
31:14			<b>Reserved Field:</b> Yes
13	NAKMsk	R/W	NAK interrupt Mask (NAKMsk) <b>Values:</b> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask NAK interrupt</li> <li>■ 0x1 (NOMASK): No NAK interrupt Mask</li> </ul> <b>Value After Reset:</b> 0x0 <b>Exists:</b> Always
12:10			<b>Reserved Field:</b> Yes

**Table 7-69 Fields for Register: DIEPEACHMSK0 (Continued)**

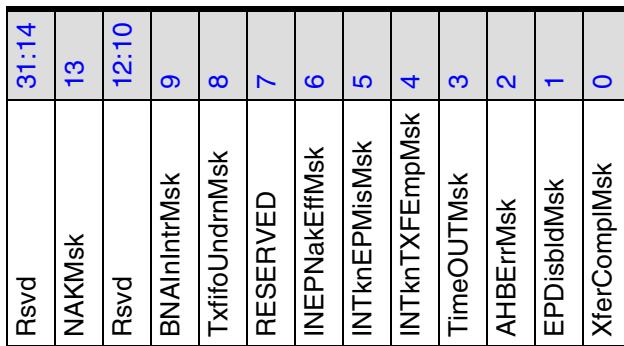
Bits	Name	Memory Access	Description
9	BNAInIntrMsk	R/W	<p>BNA interrupt Mask (BNAInIntrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask BNA interrupt</li> <li>■ 0x1 (NOMASK): No BNA interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA ==1</p>
8	TxfifoUndrnMsk	R/W	<p>Fifo Underrun Mask (TxfifoUndrnMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Fifo Underrun</li> <li>■ 0x1 (NOMASK): No Fifo Underrun Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
7	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
6	INEPNakEffMsk	R/W	<p>IN Endpoint NAK Effective Mask (INEPNakEffMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint NAK Effective</li> <li>■ 0x1 (NOMASK): No IN Endpoint NAK Effective Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
5	INTknEPMisMsk	R/W	<p>IN Token received with EP Mismatch Mask (INTknEPMisMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Token received with EP Mismatch</li> <li>■ 0x1 (NOMASK): No IN Token received with EP Mismatch Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
4	INTknTxFEmpMsk	R/W	<p>IN Token Received When TxFIFO Empty Mask (INTknTxFEmpMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Token received When TxFIFO Empty</li> <li>■ 0x1 (NOMASK): No IN Token received When TxFIFO Empty Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-69 Fields for Register: DIEPEACHMSK0 (Continued)**

Bits	Name	Memory Access	Description
3	TimeOUTMsk	R/W	<p>Timeout Condition Mask (TimeOUTMsk) (Non-isochronous endpoints)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Timeout Condition</li> <li>■ 0x1 (NOMASK): No Timeout Condition Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
2	AHBErrMsk	R/W	<p>AHB Error Mask (AHBErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask AHB Error</li> <li>■ 0x1 (NOMASK): No AHB Error Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE == 2</p>
1	EPDisbldMsk	R/W	<p>Endpoint Disabled Interrupt Mask (EPDisbldMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Endpoint Disabled Interrupt</li> <li>■ 0x1 (NOMASK): No Endpoint Disabled Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
0	XferComplMsk	R/W	<p>Transfer Completed Interrupt Mask (XferComplMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Transfer Completed Interrupt</li> <li>■ 0x1 (NOMASK): No Transfer Completed Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.65 DIEPEACHMSKi (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device Each IN Endpoint Interrupt Register
- **Description:** This register contains the interrupts for the IN Endpoints of the Device controller.  
**Note:** This register exists for an endpoint i if the OTG\_EP\_DIR\_i parameter is 0 or 1 for that endpoint.
- **Size:** 32 bits
- **Offset:** 0x840 + i\*004
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 15) && OTG\_MULTI\_PROC\_INTRPT == 1 && (OTG\_EP\_DIR\_15 != 2)



**Table 7-70 Fields for Register: DIEPEACHMSKi (for i = 1; i <= OTG\_NUM\_EPS)**

Bits	Name	Memory Access	Description
31:14			<b>Reserved Field:</b> Yes
13	NAKMsk	R/W	<b>NAK interrupt Mask (NAKMsk)</b> <b>Values:</b> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask NAK Interrupt</li> <li>■ 0x1 (NOMASK): No NAK Interrupt Mask</li> </ul> <b>Value After Reset:</b> 0x0 <b>Exists:</b> Always
12:10			<b>Reserved Field:</b> Yes
9	BNAlnIntrMsk	R/W	<b>BNA interrupt Mask (BNAlnIntrMsk)</b> <b>Values:</b> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask BNA Interrupt</li> <li>■ 0x1 (NOMASK): No BNA Interrupt Mask</li> </ul> <b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_EN_DESC_DMA ==1

**Table 7-70 Fields for Register: DIEPEACHMSKi (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

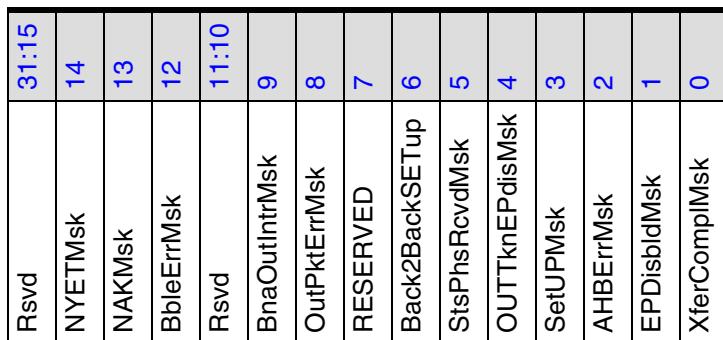
<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
8	TxfifoUndrnMsk	R/W	<p>Fifo Underrun Mask (TxfifoUndrnMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Fifo Underrun Interrupt</li> <li>■ 0x1 (NOMASK): No Fifo Underrun Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
7	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
6	INEPNakEffMsk	R/W	<p>IN Endpoint NAK Effective Mask (INEPNakEffMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Endpoint NAK Effective</li> <li>■ 0x1 (NOMASK): No IN Endpoint NAK Effective Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
5	INTknEPMisMsk	R/W	<p>IN Token received with EP Mismatch Mask (INTknEPMisMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Token received with EP Mismatch Interrupt</li> <li>■ 0x1 (NOMASK): No Mask IN Token received with EP Mismatch Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
4	INTknTXFEmpMsk	R/W	<p>IN Token Received When TxFIFO Empty Mask(INTknTXFEmpMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask IN Token Received When TxFIFO Empty Interrupt</li> <li>■ 0x1 (NOMASK): No IN Token Received When TxFIFO Empty Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-70 Fields for Register: DIEPEACHMSK<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
3	TimeOUTMsk	R/W	<p>Timeout Condition Mask (TimeOUTMsk) (Non-isochronous endpoints)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Timeout Condition Interrupt</li> <li>■ 0x1 (NOMASK): No Timeout Condition Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
2	AHBErrMsk	R/W	<p>AHB Error Mask (AHBErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask AHB Error</li> <li>■ 0x1 (NOMASK): No AHB Error Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE == 2</p>
1	EPDisbldMsk	R/W	<p>Endpoint Disabled Interrupt Mask (EPDisbldMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Endpoint Disabled Interrupt</li> <li>■ 0x1 (NOMASK): No Endpoint Disabled Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
0	XferComplMsk	R/W	<p>Transfer Completed Interrupt Mask (XferComplMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Transfer Completed Interrupt</li> <li>■ 0x1 (NOMASK): No Transfer Completed Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.66 DOEPEACHMSK0

- **Name:** Device Each OUT Endpoint 0 Interrupt Register
- **Description:** This register is available in device mode and only when parameter OTG\_MULTI\_PROC\_INTRPT=1. These registers are endpoint specific mask registers for (DOEPINTn). The OUT endpoint interrupt for a specific status in the DOEPINTn register can be masked by writing 0 to the corresponding bit in this register. Status bits are masked by default.
- **Size:** 32 bits
- **Offset:** 0x880
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 0) && OTG\_MULTI\_PROC\_INTRPT==1



**Table 7-71 Fields for Register: DOEPEACHMSK0**

Bits	Name	Memory Access	Description
31:15			<b>Reserved Field:</b> Yes
14	NYETMsk	R/W	<b>NYET interrupt Mask (NYETMsk)</b> <b>Values:</b> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask NYET Interrupt</li> <li>■ 0x1 (NOMASK): No NYET Interrupt Mask</li> </ul> <b>Value After Reset:</b> 0x0 <b>Exists:</b> Always
13	NAKMsk	R/W	<b>NAK interrupt Mask (NAKMsk)</b> <b>Values:</b> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask NAK Interrupt</li> <li>■ 0x1 (NOMASK): No NAK Interrupt Mask</li> </ul> <b>Value After Reset:</b> 0x0 <b>Exists:</b> Always

**Table 7-71 Fields for Register: DOEPEACHMSK0 (Continued)**

Bits	Name	Memory Access	Description
12	BbleErrMsk	R/W	<p>Babble Error interrupt Mask (BbleErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Babble Error Interrupt</li> <li>■ 0x1 (NOMASK): No Babble Error Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
11:10			<b>Reserved Field:</b> Yes
9	BnaOutIntrMsk	R/W	<p>BNA interrupt Mask (BnaOutIntrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask BNA Interrupt</li> <li>■ 0x1 (NOMASK): No BNA Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA ==1</p>
8	OutPktErrMsk	R/W	<p>OUT Packet Error Mask (OutPktErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Packet Error</li> <li>■ 0x1 (NOMASK): No OUT Packet Error Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
7	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
6	Back2BackSETup	R/W	<p>Back-to-Back SETUP Packets Received Mask (Back2BackSETup)</p> <p>Applies to control OUT endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Back-to-Back SETUP Packets Received</li> <li>■ 0x1 (NOMASK): No Back-to-Back SETUP Packets Received Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE !=0</p>

**Table 7-71 Fields for Register: DOEPEACHMSK0 (Continued)**

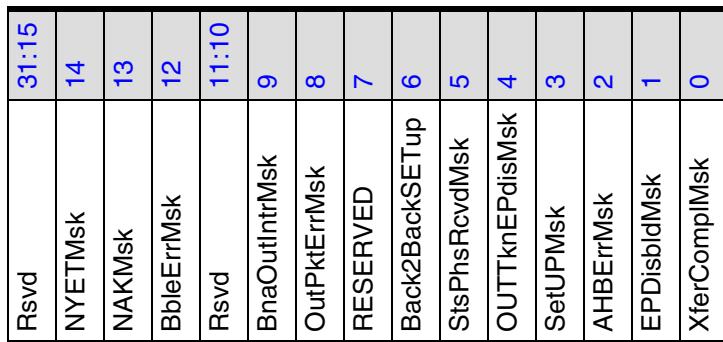
<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
5	StsPhsRcvdMsk	R/W	<p>Status Phase Received Received Mask (StsPhsRcvdMsk) Applies to control OUT endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Status Phase Received</li> <li>■ 0x1 (NOMASK): No Status Phase Received Received Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_ARCHITECTURE !=0</p>
4	OUTTknEPdisMsk	R/W	<p>OUT Token Received when Endpoint Disabled Mask (OUTTknEPdisMsk) Applies to control OUT endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Token Received when Endpoint Disabled</li> <li>■ 0x1 (NOMASK): No OUT Token Received when Endpoint Disabled Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
3	SetUPMsk	R/W	<p>SETUP Phase Done Mask (SetUPMsk) Applies to control endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask SETUP Phase Done</li> <li>■ 0x1 (NOMASK): No SETUP Phase Done Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
2	AHBErrMsk	R/W	<p>AHB Error (AHBErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask AHB Error</li> <li>■ 0x1 (NOMASK): No AHB Error Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_ARCHITECTURE == 2</p>
1	EPDisbldMsk	R/W	<p>Endpoint Disabled Interrupt Mask (EPDisbldMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Endpoint Disabled Interrupt</li> <li>■ 0x1 (NOMASK): No Endpoint Disabled Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

**Table 7-71 Fields for Register: DOEPEACHMSK0 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
0	XferComplMsk	R/W	<p>Transfer Completed Interrupt Mask (XferComplMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Transfer Completed Interrupt</li> <li>■ 0x1 (NOMASK): No Transfer Completed Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.67 DOEPEACHMSKi (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device Each OUT Endpoint Interrupt Register
- **Description:** This register contains the interrupts for the OUT Endpoints of the Device controller.  
**Note:** This register exists for an endpoint i if the OTG\_EP\_DIR\_i parameter is 0 or 1 for that endpoint.
- **Size:** 32 bits
- **Offset:** 0x880 + i\*004
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 15) && OTG\_MULTI\_PROC\_INTRPT==1 && (OTG\_EP\_DIR\_15 !=1)



**Table 7-72 Fields for Register: DOEPEACHMSKi (for i = 1; i <= OTG\_NUM\_EPS)**

Bits	Name	Memory Access	Description
31:15			<b>Reserved Field:</b> Yes
14	NYETMsk	R/W	<p>NYET interrupt Mask (NYETMsk)  <b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask NYET Interrupt</li> <li>■ 0x1 (NOMASK): No NYET Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
13	NAKMsk	R/W	<p>NAK interrupt Mask (NAKMsk)  <b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask NAK Response Received Interrupt</li> <li>■ 0x1 (NOMASK): No NAK Response Received Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-72 Fields for Register: DOEPEACHMSK<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
12	BbleErrMsk	R/W	<p>Babble Error interrupt Mask (BbleErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Babble Error Interrupt</li> <li>■ 0x1 (NOMASK): No Babble Error Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
11:10			<b>Reserved Field:</b> Yes
9	BnaOutIntrMsk	R/W	<p>BNA interrupt Mask (BnaOutIntrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask BNA Interrupt</li> <li>■ 0x1 (NOMASK): No BNA Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DESC_DMA ==1</p>
8	OutPktErrMsk	R/W	<p>OUT Packet Error Mask (OutPktErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Packet Error Interrupt</li> <li>■ 0x1 (NOMASK): No OUT Packet Error Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
7	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
6	Back2BackSETup	R/W	<p>Back-to-Back SETUP Packets Received Mask (Back2BackSETup)</p> <p>Applies to control OUT endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Back-to-Back SETUP Packets Received</li> <li>■ 0x1 (NOMASK): No Back-to-Back SETUP Packets Received Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE !=0</p>

**Table 7-72 Fields for Register: DOEPEACHMSKi (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

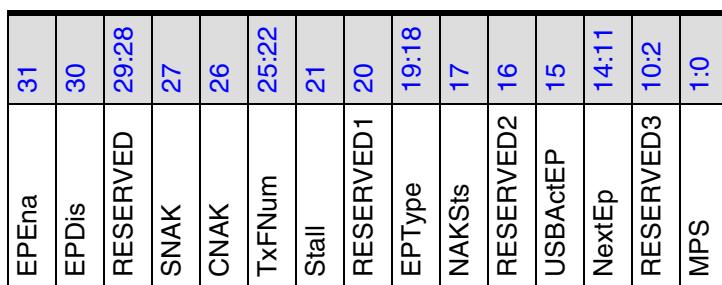
<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
5	StsPhsRcvdMsk	R/W	<p>Status Phase Received Mask (StsPhsRcvdMsk) Applies to control OUT endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Status Phase Received</li> <li>■ 0x1 (NOMASK): No Status Phase Received Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_ARCHITECTURE !=0</p>
4	OUTTknEPdisMsk	R/W	<p>OUT Token Received when Endpoint Disabled Mask (OUTTknEPdisMsk) Applies to control OUT endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask OUT Token Received when Endpoint Disabled Interrupt</li> <li>■ 0x1 (NOMASK): No OUT Token Received when Endpoint Disabled Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
3	SetUPMsk	R/W	<p>SETUP Phase Done Mask (SetUPMsk) Applies to control endpoints only.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask SETUP Phase Done Interrupt</li> <li>■ 0x1 (NOMASK): No SETUP Phase Done Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
2	AHBErrMsk	R/W	<p>AHB Error (AHBErrMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): AHB Error Mask</li> <li>■ 0x1 (NOMASK): No AHB Error Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_ARCHITECTURE == 2</p>
1	EPDisbldMsk	R/W	<p>Endpoint Disabled Interrupt Mask (EPDisbldMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Mask Endpoint Disabled Interrupt</li> <li>■ 0x1 (NOMASK): No Endpoint Disabled Interrupt Mask</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

**Table 7-72 Fields for Register: DOEPEACHMSKi (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
0	XferComplMsk	R/W	<p>Transfer Completed Interrupt Mask (XferComplMsk)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (MASK): Transfer Completed Mask</li> <li>■ 0x1 (NOMASK): No Transfer Completed Mask</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.68 DIEPCTL0

- **Name:** Device Control IN Endpoint 0 Control Register
- **Description:** This register is used to control the characteristics of the IN Endpoint 0 of the Device controller.
- **Size:** 32 bits
- **Offset:** 0x900
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6)



**Table 7-73 Fields for Register: DIEPCTL0**

Bits	Name	Memory Access	Description
31	EPEna	R/W1S	<p>Endpoint Enable (EPEna) 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:</p> <ul style="list-style-type: none"> <li>■ Endpoint Disabled</li> <li>■ Transfer Completed</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No action</li> <li>■ 0x1 (ACTIVE): Enable Endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-73 Fields for Register: DIEPCTL0 (Continued)**

Bits	Name	Memory Access	Description
30	EPDis	R/W1S	<p>Endpoint Disable (EPDis)  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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No action</li> <li>■ 0x1 (ACTIVE): Disabled Endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
29:28	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> 0  <b>Testable:</b> writeAsRead  <b>Write Constraint:</b> writeAsRead</p>
27	SNAK	W	<p>Set NAK (SNAK) 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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOSET): No action</li> <li>■ 0x1 (SET): Set NAK</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
26	CNAK	W	<p>Clear NAK (CNAK) A write to this bit clears the NAK bit for the endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOCLEAR): No action</li> <li>■ 0x1 (CLEAR): Clear NAK</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-73 Fields for Register: DIEPCTL0 (Continued)**

Bits	Name	Memory Access	Description
25:22	TxFNum	OTG_EN _DED_T X_FIFO == 1 ? R/W : R	<p>TxFIFO Number (TxFNum)</p> <ul style="list-style-type: none"> <li>■ 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.</li> <li>■ For Dedicated FIFO operation, this value is set to the FIFO number that is assigned to IN Endpoint.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (TXFIFO0): Tx FIFO 0</li> <li>■ 0x1 (TXFIFO1): Tx FIFO 1</li> <li>■ 0x2 (TXFIFO2): Tx FIFO 2</li> <li>■ 0x3 (TXFIFO3): Tx FIFO 3</li> <li>■ 0x4 (TXFIFO4): Tx FIFO 4</li> <li>■ 0x5 (TXFIFO5): Tx FIFO 5</li> <li>■ 0x6 (TXFIFO6): Tx FIFO 6</li> <li>■ 0x7 (TXFIFO7): Tx FIFO 7</li> <li>■ 0x8 (TXFIFO8): Tx FIFO 8</li> <li>■ 0x9 (TXFIFO9): Tx FIFO 9</li> <li>■ 0xa (TXFIFO10): Tx FIFO 10</li> <li>■ 0xb (TXFIFO11): Tx FIFO 11</li> <li>■ 0xc (TXFIFO12): Tx FIFO 12</li> <li>■ 0xd (TXFIFO13): Tx FIFO 13</li> <li>■ 0xe (TXFIFO14): Tx FIFO 14</li> <li>■ 0xf (TXFIFO15): Tx FIFO 15</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
21	Stall	R/W1S	<p>STALL Handshake (Stall)</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 Nonperiodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Stall</li> <li>■ 0x1 (ACTIVE): Stall Handshake</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-73 Fields for Register: DIEPCTL0 (Continued)**

Bits	Name	Memory Access	Description
20	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
19:18	EPType	R	<p>Endpoint Type (EPType) Hardcoded to 00 for control.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ACTIVE): Endpoint Control 0</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
17	NAKsts	R	<p>NAK Status (NAKsts) Indicates the following:</p> <ul style="list-style-type: none"> <li>■ 1'b0: The core is transmitting non-NAK handshakes based on the FIFO status</li> <li>■ 1'b1: The core is transmitting NAK handshakes on this endpoint. 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.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): The core is transmitting non-NAK handshakes based on the FIFO status</li> <li>■ 0x1 (ACTIVE): The core is transmitting NAK handshakes on this endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>
16	RESERVED2	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-73 Fields for Register: DIEPCTL0 (Continued)**

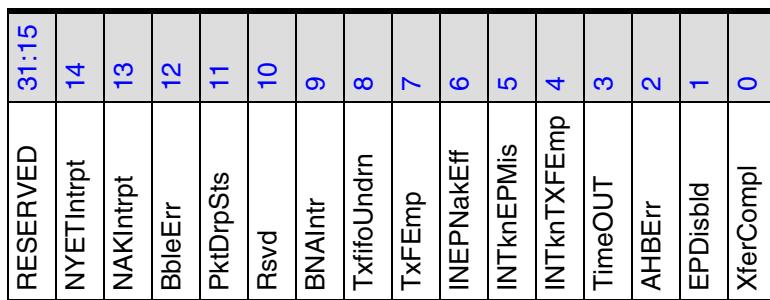
<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
15	USBActEP	R	<p>USB Active Endpoint (USBActEP) This bit is always SET to 1, indicating that control endpoint 0 is always active in all configurations and interfaces.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (ACTIVE0): Control endpoint is always active</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p>
14:11	NextEp	R/W	<p>Next Endpoint (NextEp) 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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (EP0): EP0</li> <li>■ 0x1 (EP1): EP1</li> <li>■ 0x2 (EP2): EP2</li> <li>■ 0x3 (EP3): EP3</li> <li>■ 0x4 (EP4): EP4</li> <li>■ 0x5 (EP5): EP5</li> <li>■ 0x6 (EP6): EP6</li> <li>■ 0x7 (EP7): EP7</li> <li>■ 0x8 (EP8): EP8</li> <li>■ 0x9 (EP9): EP9</li> <li>■ 0xa (EP10): EP10</li> <li>■ 0xb (EP11): EP11</li> <li>■ 0xc (EP12): EP12</li> <li>■ 0xd (EP13): EP13</li> <li>■ 0xe (EP14): EP14</li> <li>■ 0xf (EP15): EP15</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE !=0 &amp;&amp; OTG_EN_DED_TX_FIFO == 0</p>

**Table 7-73 Fields for Register: DIEPCTL0 (Continued)**

Bits	Name	Memory Access	Description
10:2	RESERVED3	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
1:0	MPS	R/W	<p>Maximum Packet Size (MPS) Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint.</p> <ul style="list-style-type: none"> <li>■ 2'b00: 64 bytes</li> <li>■ 2'b01: 32 bytes</li> <li>■ 2'b10: 16 bytes</li> <li>■ 2'b11: 8 bytes</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (BYTES64): 64 bytes</li> <li>■ 0x1 (BYTES32): 32 bytes</li> <li>■ 0x2 (BYTES16): 16 bytes</li> <li>■ 0x3 (BYTES8): 8 bytes</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.69 DIEPINT0

- **Name:** Device IN Endpoint 0 Interrupt Register
- **Description:** This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in the "Interrupt Hierarchy" figure in the databook. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers
- **Size:** 32 bits
- **Offset:** 0x908
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6)



**Table 7-74 Fields for Register: DIEPINT0**

Bits	Name	Memory Access	Description
31:15	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
14	NYETInrpt	R/W1C	<p>NYET Interrupt (NYETInrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No interrupt</li> <li>■ 0x1 (ACTIVE): NYET Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-74 Fields for Register: DIEPINT0 (Continued)**

Bits	Name	Memory Access	Description
13	NAKIntrpt	R/W1C	<p>NAK Interrupt (NAKIntrpt)  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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No interrupt</li> <li>■ 0x1 (ACTIVE): NAK Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
12	BbleErr	R/W1C	<p>NAK Interrupt (BbleErr)  The core generates this interrupt when babble is received for the endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No interrupt</li> <li>■ 0x1 (ACTIVE): BbleErr interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
11	PktDrpSts	R/W1C	<p>Packet Drop Status (PktDrpSts)  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.</p> <p>Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No interrupt</li> <li>■ 0x1 (ACTIVE): Packet Drop Status</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
10			<b>Reserved Field:</b> Yes
9	BNAIntr	R/W1C	<p>BNA (Buffer Not Available) Interrupt (BNAIntr)  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, such as Host busy or DMA done.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No BNA interrupt</li> <li>■ 0x1 (ACTIVE): BNA interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-74 Fields for Register: DIEPINT0 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
8	TxfifoUndrn	R/W1C	<p>Fifo Underrun (TxfifoUndrn) Applies to IN endpoints only. The core generates this interrupt when it detects a transmit FIFO underrun condition in threshold mode for this endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Fifo Underrun interrupt</li> <li>■ 0x1 (ACTIVE): Fifo Underrun interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DED_TX_FIFO == 1</p>
7	TxFEmp	R	<p>Transmit FIFO Empty (TxFEmp) 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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Transmit FIFO Empty interrupt</li> <li>■ 0x1 (ACTIVE): Transmit FIFO Empty interrupt</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>
6	INEPNakEff	R/W1C	<p>IN Endpoint NAK Effective (INEPNakEff) 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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No IN Endpoint NAK Effective interrupt</li> <li>■ 0x1 (ACTIVE): IN Endpoint NAK Effective interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-74 Fields for Register: DIEPINT0 (Continued)**

Bits	Name	Memory Access	Description
5	INTknEPMis	R/W1C	<p>IN Token Received with EP Mismatch (INTknEPMis) 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><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No IN Token Received with EP Mismatch interrupt</li> <li>■ 0x1 (ACTIVE): IN Token Received with EP Mismatch interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
4	INTknTXFEmp	R/W1C	<p>IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No IN Token Received when TxFIFO Empty interrupt</li> <li>■ 0x1 (ACTIVE): IN Token Received when TxFIFO Empty Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
3	TimeOUT	R/W1C	<p>Timeout Condition (TimeOUT)</p> <ul style="list-style-type: none"> <li>■ In shared TX FIFO mode, applies to non-isochronous IN endpoints only.</li> <li>■ In dedicated FIFO mode, applies only to Control IN endpoints.</li> <li>■ In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted.</li> </ul> <p>Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Timeout interrupt</li> <li>■ 0x1 (ACTIVE): Timeout interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-74 Fields for Register: DIEPINT0 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
2	AHBErr	R/W1C	<p>AHB Error (AHBErr) Applies to IN and OUT endpoints.</p> <p>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. For details, see "AHB Error Handling" in the Programming Guide.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No AHB Error Interrupt</li> <li>■ 0x1 (ACTIVE): AHB Error interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE == 2</p>
1	EPDisbld	R/W1C	<p>Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints.</p> <p>This bit indicates that the endpoint is disabled per the application's request.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Endpoint Disabled Interrupt</li> <li>■ 0x1 (ACTIVE): Endpoint Disabled Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-74 Fields for Register: DIEPINT0 (Continued)**

Bits	Name	Memory Access	Description
0	XferCompl	R/W1C	<p>Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints.</p> <ul style="list-style-type: none"> <li>■ When Scatter/Gather DMA mode is enabled           <ul style="list-style-type: none"> <li>- For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO.</li> <li>- 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.</li> </ul> </li> <li>■ 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.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Transfer Complete Interrupt</li> <li>■ 0x1 (ACTIVE): Transfer Completed Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.70 DIEPTSIZ0

- **Name:** Device IN Endpoint 0 Transfer Size Register
- **Description:** The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using Endpoint Enable bit of the Device Control Endpoint 0 Control registers (DIEPCTL0.EPEna/DOEPCTL0.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit. Nonzero endpoints use the registers for endpoints 115. When Scatter/Gather DMA mode is enabled, this register must not be programmed by the application. If the application reads this register when Scatter/Gather DMA mode is enabled, the core returns all zeros.
- **Size:** 32 bits
- **Offset:** 0x910
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6)



**Table 7-75 Fields for Register: DIEPTSIZ0**

Bits	Name	Memory Access	Description
31:21	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
20:19	PktCnt	R/W	<p>Packet Count (PktCnt)</p> <p>Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.</p> <ul style="list-style-type: none"> <li>■ IN Endpoints : This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.</li> <li>■ OUT Endpoints: This field is decremented every time a packet (maximum size or short packet) is written to the RxFIFO.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-75 Fields for Register: DIEPTSIZE0 (Continued)**

Bits	Name	Memory Access	Description
18:7	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
6:0	XferSize	R/W	<p>Transfer Size (XferSize)</p> <p>This field contains the transfer size in bytes for the current endpoint. The transfer size (XferSize) = Sum of buffer sizes across all descriptors in the list for the endpoint. In Buffer DMA, 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.</p> <ul style="list-style-type: none"> <li>■ IN Endpoints: The core decrements this field every time a packet from the external memory is written to the TxFIFO.</li> <li>■ OUT Endpoints: The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.71 DIEPDMA0

- **Name:** Device IN Endpoint 0 DMA Address Register
- **Description:** This register contains the DMA Address for the IN Endpoint 0 of the Device controller.
- **Size:** 32 bits
- **Offset:** 0x914
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6 && OTG\_ARCHITECTURE == 2)



**Table 7-76 Fields for Register: DIEPDMA0**

Bits	Name	Memory Access	Description
31:0	DMAAddr	R/W	<p><b>DMAAddr</b></p> <p>This field holds the start address of the external memory for storing or fetching endpoint data.</p> <p>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.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <p>When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</p> <p>When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</p> <p><b>Value After Reset:</b> DMA registers are implemented in SPRAM. The reset values of these registers in simulation can be 0/X depending upon the SPRAM model.</p> <p><b>Exists:</b> Always</p>

### 7.1.72 DTXFSTS0

- **Name:** Device IN Endpoint Transmit FIFO Status Register 0
- **Description:** This register contains information about the IN Endpoint Transmit FIFO of the Device controller.
- **Size:** 32 bits
- **Offset:** 0x918
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6)



**Table 7-77 Fields for Register: DTXFSTS0**

Bits	Name	Memory Access	Description
31:16	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-77 Fields for Register: DTXFSTS0 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
15:0	INEPTxFSpAvail	R	<p>IN Endpoint TxFIFO Space Avail (INEPTxFSpAvail)      Indicates the amount of free space available in the Endpoint TxFIFO.      Values are in terms of 32-bit words.</p> <ul style="list-style-type: none"> <li>■ 16'h0: Endpoint TxFIFO is full</li> <li>■ 16'h1: 1 word available</li> <li>■ 16'h2: 2 words available</li> <li>■ 16'hn: n words available (where 0 &lt; n &lt; 32,768)</li> <li>■ 16'h8000: 32,768 words available</li> <li>■ Others: Reserved</li> </ul> <p>In DRD configurations (OTG_MODE = 0, 1, or 2) with dynamic FIFO sizing feature enabled (OTG_DFIFO_DYNAMIC = 1), the value of this field is,</p> <ul style="list-style-type: none"> <li>■ the maximum value of (OTG_TX_HNPERIO_DFIFO_DEPTH, OTG_TX_DINEP_DFIFO_DEPTH_0) during reset, and</li> <li>■ OTG_TX_DINEP_DFIFO_DEPTH_0, immediately after reset deassertion</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always  <b>Testable:</b> untestable</p>

### 7.1.73 DIEPDMAB0

- **Name:** Device IN Endpoint 0 Buffer Address Register
- **Description:** This register contains the DMA Buffer Address for the IN Endpoint 0 of the Device controller.
- **Size:** 32 bits
- **Offset:** 0x91c
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && OTG\_EN\_DESC\_DMA==1



**Table 7-78 Fields for Register: DIEPDMAB0**

Bits	Name	Memory Access	Description
31:0	DMABufferAddr	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p> <p><b>Value After Reset:</b> DMA registers are implemented in SPRAM. The reset values of these registers in simulation can be 0/X depending upon the SPRAM model.</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> untestable</p>

### 7.1.74 DIEPCTL*i* (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device Control IN Endpoint Control Register
- **Description:** This register is used to control the characteristics of Endpoint i.  
**Note:** This register exists for an endpoint i if the OTG\_EP\_DIR\_i parameter is 0 or 1 for that endpoint.
- **Size:** 32 bits
- **Offset:** 0x900 + i\*20
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 1) && OTG\_EP\_DIR\_1 != 2

31	EPEna	30	EPDis	29	SetD1PID	28	SetD0PID	27	SNAK	26	CNAK	25:22	TxFNlm	21	Stall	20	Rsvd	19:18	EPType	17	NAKSts	16	DPID	15	USBActEP	14:11	NextEp	10:0	MPS
----	-------	----	-------	----	----------	----	----------	----	------	----	------	-------	--------	----	-------	----	------	-------	--------	----	--------	----	------	----	----------	-------	--------	------	-----

**Table 7-79 Fields for Register: DIEPCTL<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
31	EPEna	R/W1S	<p>Endpoint Enable (EPEna) Applies to IN and OUT endpoints.</p> <ul style="list-style-type: none"> <li>■ When Scatter/Gather DMA mode is enabled,           <ul style="list-style-type: none"> <li>- For IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup.</li> <li>- For OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup.</li> </ul> </li> <li>■ When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode:           <ul style="list-style-type: none"> <li>- For IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint.</li> <li>- For OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB.</li> </ul> </li> <li>■ The core clears this bit before setting any of the following interrupts on this endpoint:           <ul style="list-style-type: none"> <li>- SETUP Phase Done</li> <li>- Endpoint Disabled</li> <li>- Transfer Completed</li> </ul> </li> </ul> <p>Note: For control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Action</li> <li>■ 0x1 (ACTIVE): Enable Endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
30	EPDis	R/W1S	<p>Endpoint Disable (EPDis) Applies to IN and OUT endpoints.</p> <p>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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Action</li> <li>■ 0x1 (ACTIVE): Disable Endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-79 Fields for Register: DIEPCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
29	SetD1PID	W	<p>SetD1PID</p> <ul style="list-style-type: none"> <li>■ Set DATA1 PID (SetD1PID) <ul style="list-style-type: none"> <li>- Applies to interrupt and bulk IN and OUT endpoints only.</li> <li>- Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1.</li> <li>- This field is applicable both for Scatter-Gather DMA mode and non Scatter-Gather DMA mode.</li> </ul> </li> <li>■ Set odd (micro)Frame (SetOddFr) <ul style="list-style-type: none"> <li>- Applies to isochronous IN and OUT endpoints only.</li> <li>- Writing to this field sets the even and odd (micro)Frame (EO_FrNum) field to odd (micro)Frame.</li> <li>- This field is not applicable for Scatter-Gather DMA mode.</li> </ul> </li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disables Set DATA1 PID</li> <li>■ 0x1 (ENABLED): Enables Set DATA1 PID</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
28	SetD0PID	W	<p>SetD0PID</p> <ul style="list-style-type: none"> <li>■ Set DATA0 PID (SetD0PID) <ul style="list-style-type: none"> <li>- Applies to interrupt/bulk IN and OUT endpoints only.</li> <li>- Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</li> <li>- This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</li> </ul> </li> </ul> <p>In non-Scatter/Gather DMA mode: Set Even (micro)Frame (SetEvenFr)</p> <ul style="list-style-type: none"> <li>-- Applies to isochronous IN and OUT endpoints only.</li> <li>-- 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.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disables Set DATA0 PID</li> <li>■ 0x1 (ENABLED): Endpoint Data PID to DATA0</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-79 Fields for Register: DIEPCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
27	SNAK	W	<p>Set NAK (SNAK)  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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Set NAK</li> <li>■ 0x1 (ACTIVE): Set NAK</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
26	CNAK	W	<p>Clear NAK (CNAK)  A write to this bit clears the NAK bit for the endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Clear NAK</li> <li>■ 0x1 (ACTIVE): Clear NAK</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-79 Fields for Register: DIEPCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
25:22	TxFNum	R/W	<p>TxFIFO Number (TxFNum)  Shared FIFO Operation non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number.</p> <ul style="list-style-type: none"> <li>■ 4'h0: Non-Periodic TxFIFO</li> <li>■ Others: Specified Periodic TxFIFO.number</li> </ul> <p>Note: An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Otherwise, a separate periodic FIFO must be allocated for an interrupt IN endpoint, and the number of this FIFO must be programmed into the TxFNum field. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area.</p> <p><i>Dedicated FIFO Operation:</i> 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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (TXFIFO0): Tx FIFO 0</li> <li>■ 0x1 (TXFIFO1): Tx FIFO 1</li> <li>■ 0x2 (TXFIFO2): Tx FIFO 2</li> <li>■ 0x3 (TXFIFO3): Tx FIFO 3</li> <li>■ 0x4 (TXFIFO4): Tx FIFO 4</li> <li>■ 0x5 (TXFIFO5): Tx FIFO 5</li> <li>■ 0x6 (TXFIFO6): Tx FIFO 6</li> <li>■ 0x7 (TXFIFO7): Tx FIFO 7</li> <li>■ 0x8 (TXFIFO8): Tx FIFO 8</li> <li>■ 0x9 (TXFIFO9): Tx FIFO 9</li> <li>■ 0xa (TXFIFO10): Tx FIFO 10</li> <li>■ 0xb (TXFIFO11): Tx FIFO 11</li> <li>■ 0xc (TXFIFO12): Tx FIFO 12</li> <li>■ 0xd (TXFIFO13): Tx FIFO 13</li> <li>■ 0xe (TXFIFO14): Tx FIFO 14</li> <li>■ 0xf (TXFIFO15): Tx FIFO 15</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-79 Fields for Register: DIEPCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
21	Stall	R/W1S	<p>STALL Handshake (Stall) 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. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): STALL All non-active tokens</li> <li>■ 0x1 (ACTIVE): STALL All Active Tokens</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
20			<b>Reserved Field:</b> Yes
19:18	EPType	R/W	<p>Endpoint Type (EPType) This is the transfer type supported by this logical endpoint.</p> <ul style="list-style-type: none"> <li>■ 2'b00: Control</li> <li>■ 2'b01: Isochronous</li> <li>■ 2'b10: Bulk</li> <li>■ 2'b11: Interrupt</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (CONTROL): Control</li> <li>■ 0x1 (ISOCHRONOUS): Isochronous</li> <li>■ 0x2 (BULK): Bulk</li> <li>■ 0x3 (INTERRUPT): Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-79 Fields for Register: DIEPCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
17	NAKSts	R	<p>NAK Status (NAKSts) Indicates the following:</p> <ul style="list-style-type: none"> <li>■ 1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</li> <li>■ 1'b1: The core is transmitting NAK handshakes on this endpoint.</li> </ul> <p>When either the application or the core sets this bit:</p> <ul style="list-style-type: none"> <li>■ The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.</li> <li>■ For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.</li> <li>■ For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.</li> </ul> <p>Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NONNAK): The core is transmitting non-NAK handshakes based on the FIFO status</li> <li>■ 0x1 (NAK): The core is transmitting NAK handshakes on this endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always <b>Testable:</b> readOnly</p>

**Table 7-79 Fields for Register: DIEPCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
16	DPID	R	<p><i>Endpoint Data PID (DPID)</i>      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> <ul style="list-style-type: none"> <li>■ 1'b0: DATA0</li> <li>■ 1'b1: DATA1</li> </ul> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p>Even/Odd (Micro)Frame (EO_FrNum)      In non-Scatter/Gather DMA mode:      Applies to isochronous IN and OUT endpoints only.      Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro)frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Even (micro)frame</li> <li>■ 1'b1: Odd (micro)frame</li> </ul> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DATA0EVENFRM): DATA0 or Even Frame</li> <li>■ 0x1 (DATA1ODDFRM): DATA1 or Odd Frame</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>

**Table 7-79 Fields for Register: DIEPCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
15	USBActEP	R/W	<p>USB Active Endpoint (USBActEP)            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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Not Active</li> <li>■ 0x1 (ENABLED): USB Active Endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
14:11	NextEp	R/W	<p>Next Endpoint (NextEp)            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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (EP0): EP0</li> <li>■ 0x1 (EP1): EP1</li> <li>■ 0x2 (EP2): EP2</li> <li>■ 0x3 (EP3): EP3</li> <li>■ 0x4 (EP4): EP4</li> <li>■ 0x5 (EP5): EP5</li> <li>■ 0x6 (EP6): EP6</li> <li>■ 0x7 (EP7): EP7</li> <li>■ 0x8 (EP8): EP8</li> <li>■ 0x9 (EP9): EP9</li> <li>■ 0xa (EP10): EP10</li> <li>■ 0xb (EP11): EP11</li> <li>■ 0xc (EP12): EP12</li> <li>■ 0xd (EP13): EP13</li> <li>■ 0xe (EP14): EP14</li> <li>■ 0xf (EP15): EP15</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE !=0 &amp;&amp; OTG_EN_DED_TX_FIFO == 0</p>

**Table 7-79 Fields for Register: DIEPCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
10:0	MPS	R/W	<p>Maximum Packet Size (MPS)            The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always  <b>Testable:</b> writeAsRead</p>

### 7.1.75 DIEPINTi (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device IN Endpoint Interrupt Register
- **Description:** This register contains the interrupts for the IN Endpoint i of the Device controller.
- Note:** This register exists for an endpoint i if the OTG\_EP\_DIR\_i parameter is 0 or 1 for that endpoint.
- **Size:** 32 bits
- **Offset:** 0x908 + i\*20
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 1) && (OTG\_EP\_DIR\_1 != 2)

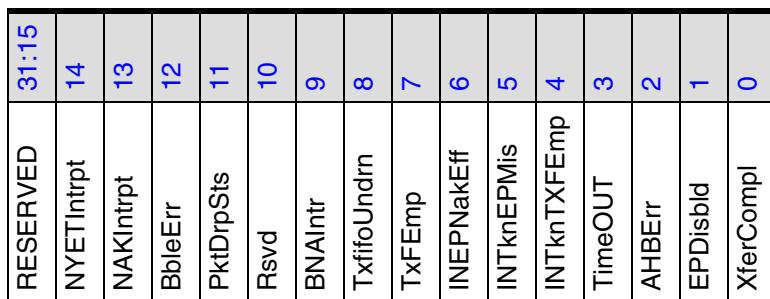


Table 7-80 Fields for Register: DIEPINTi (for i = 1; i <= OTG\_NUM\_EPS)

Bits	Name	Memory Access	Description
31:15	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
14	NYETInrpt	R/W1C	<p>NYET Interrupt (NYETInrpt)</p> <p>The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No NYET interrupt</li> <li>■ 0x1 (ACTIVE): NYET Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-80 Fields for Register: DIEPINTi (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
13	NAKIntrpt	R/W1C	<p>NAK Interrupt (NAKIntrpt)  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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No NAK interrupt</li> <li>■ 0x1 (ACTIVE): NAK Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
12	BbleErr	R/W1C	<p>NAK Interrupt (BbleErr)  The core generates this interrupt when babble is received for the endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No interrupt</li> <li>■ 0x1 (ACTIVE): BbleErr interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
11	PktDrpSts	R/W1C	<p>Packet Drop Status (PktDrpSts)  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.</p> <p>Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No interrupt</li> <li>■ 0x1 (ACTIVE): Packet Drop Status interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
10			<b>Reserved Field:</b> Yes
9	BNAIntr	R/W1C	<p>BNA (Buffer Not Available) Interrupt (BNAIntr)  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, such as Host busy or DMA done.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No BNA interrupt</li> <li>■ 0x1 (ACTIVE): BNA interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-80 Fields for Register: DIEPINTi (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
8	TxfifoUndrn	R/W1C	<p>Fifo Underrun (TxfifoUndrn) Applies to IN endpoints Only This bit is valid only If thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Tx FIFO Underrun interrupt</li> <li>■ 0x1 (ACTIVE): Tx FIFO Underrun interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENDED_TX_FIFO == 1</p>
7	TxFEmp	R	<p>Transmit FIFO Empty (TxFEmp) This bit is valid only for IN endpoints This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Transmit FIFO Empty interrupt</li> <li>■ 0x1 (ACTIVE): Transmit FIFO Empty interrupt</li> </ul> <p><b>Value After Reset:</b> 0x1</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>
6	INEPNakEff	R/W1C	<p>IN Endpoint NAK Effective (INEPNakEff) 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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Endpoint NAK Effective interrupt</li> <li>■ 0x1 (ACTIVE): IN Endpoint NAK Effective interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-80 Fields for Register: DIEPINTi (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
5	INTknEPMis	R/W1C	<p>IN Token Received with EP Mismatch (INTknEPMis) 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><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No IN Token Received with EP Mismatch interrupt</li> <li>■ 0x1 (ACTIVE): IN Token Received with EP Mismatch interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
4	INTknTXFEmp	R/W1C	<p>IN Token Received When TxFIFO is Empty (INTknTXFEmp) Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No IN Token Received interrupt</li> <li>■ 0x1 (ACTIVE): IN Token Received Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
3	TimeOUT	R/W1C	<p>Timeout Condition (TimeOUT)</p> <ul style="list-style-type: none"> <li>■ In shared TX FIFO mode, applies to non-isochronous IN endpoints only.</li> <li>■ In dedicated FIFO mode, applies only to Control IN endpoints.</li> <li>■ In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted.</li> </ul> <p>Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Timeout interrupt</li> <li>■ 0x1 (ACTIVE): Timeout interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-80 Fields for Register: DIEPINT<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

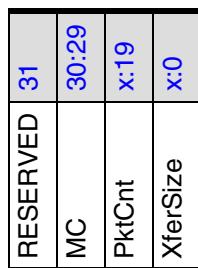
<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
2	AHBErr	R/W1C	<p>AHB Error (AHBErr) Applies to IN and OUT endpoints.</p> <p>This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. For details, see "AHB Error Handling" in the Programming Guide.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No AHB Error Interrupt</li> <li>■ 0x1 (ACTIVE): AHB Error interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE == 2</p>
1	EPDisbld	R/W1C	<p>Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints.</p> <p>This bit indicates that the endpoint is disabled per the application's request.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Endpoint Disabled Interrupt</li> <li>■ 0x1 (ACTIVE): Endpoint Disabled Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> writeAsRead</p>

**Table 7-80 Fields for Register: DIEPINTi (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
0	XferCompl	R/W1C	<p>Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints.</p> <ul style="list-style-type: none"> <li>■ When Scatter/Gather DMA mode is enabled             <ul style="list-style-type: none"> <li>- For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO.</li> <li>- 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.</li> </ul> </li> <li>■ 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.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Transfer Complete Interrupt</li> <li>■ 0x1 (ACTIVE): Transfer Complete Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.76 DIEPTSI<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device IN Endpoint Transfer Size Register
- **Description:** This register reflects the Transfer Size of the IN Endpoint i of the Device controller.  
**Note:** This register exists for an endpoint i if the OTG\_EP\_DIR\_i parameter is 0 or 1 for that endpoint.
- **Size:** 32 bits
- **Offset:** 0x910 + i\*20
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 1) && (OTG\_EP\_DIR\_1 != 2)



**Table 7-81 Fields for Register: DIEPTSI<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)**

Bits	Name	Memory Access	Description
31	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-81 Fields for Register: DIEPTSIZi (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
30:29	MC	R/W	<p>MC 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> <ul style="list-style-type: none"> <li>■ 2'b01: 1 packet</li> <li>■ 2'b10: 2 packets</li> <li>■ 2'b11: 3 packets</li> </ul> <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)</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (PACKETONE): 1 packet</li> <li>■ 0x2 (PACKETTWO): 2 packets</li> <li>■ 0x3 (PACKETTHREE): 3 packets</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
x:19	PktCnt	R/W	<p>Packet Count (PktCnt) Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Range Variable[x]:</b> OTG_PACKET_COUNT_WIDTH + 18</p>
x:0	XferSize	R/W	<p>Transfer Size (XferSize) Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only 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. The core decrements this field every time a packet from the external memory is written to the TxFIFO.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Range Variable[x]:</b> OTG_TRANS_COUNT_WIDTH - 1</p>

### 7.1.77 DIEPDMAi (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device IN Endpoint DMA Address Register
- **Description:** This register contains the DMA Address for the IN Endpoint i of the Device controller.  
**Note:** This register exists for an endpoint i if the OTG\_EP\_DIR\_i parameter is 0 or 1 for that endpoint.
- **Size:** 32 bits
- **Offset:** 0x914 + i\*20
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 1) && (OTG\_ARCHITECTURE == 2) && (OTG\_EP\_DIR\_1 != 2)



**Table 7-82 Fields for Register: DIEPDMAi (for i = 1; i <= OTG\_NUM\_EPS)**

Bits	Name	Memory Access	Description
31:0	DMAAddr	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>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.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <ul style="list-style-type: none"> <li>■ When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</li> <li>■ When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</li> </ul> <p><b>Value After Reset:</b> DMA registers are implemented in SPRAM. The reset values of these registers in simulation can be 0/X depending upon the SPRAM model.</p> <p><b>Exists:</b> Always</p>

### 7.1.78 DTXFSTS<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device IN Endpoint Transmit FIFO Status Register
  - **Description:** This register reflects the status of the IN Endpoint Transmit FIFO Status Register i of the Device controller.
- Note:** This register exists for an endpoint i if the OTG\_EP\_DIR\_i parameter is 0 or 1 for that endpoint.
- **Size:** 32 bits
  - **Offset:** 0x918 + i\*20
  - **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 1) && (OTG\_EP\_DIR\_1 != 2)



**Table 7-83 Fields for Register: DTXFSTS<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)**

Bits	Name	Memory Access	Description
31:16	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-83 Fields for Register: DTXFSTS<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
15:0	INEPTxFSpAvail	R	<p>IN Endpoint Tx FIFO Space Avail (INEPTxFSpAvail)      Indicates the amount of free space available in the Endpoint Tx FIFO.      Values are in terms of 32-bit words.</p> <ul style="list-style-type: none"> <li>■ 16'h0: Endpoint Tx FIFO is full</li> <li>■ 16'h1: 1 word available</li> <li>■ 16'h2: 2 words available</li> <li>■ 16'hn: n words available (where 0 &lt; n &lt; 32,768)</li> <li>■ 16'h8000: 32,768 words available</li> <li>■ Others: Reserved</li> </ul> <p>In DRD configurations (OTG_MODE = 0, 1, or 2) with dynamic fifo sizing feature enabled (OTG_DFIFO_DYNAMIC=1), the value of this field is,</p> <ul style="list-style-type: none"> <li>■ the maximum value of (OTG_TX_HNPERIO_DFIFO_DEPTH, OTG_TX_DINEP_DFIFO_DEPTH_0) during reset, and</li> <li>■ OTG_TX_DINEP_DFIFO_DEPTH_0, immediately after reset deassertion</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always  <b>Testable:</b> untestable</p>

### 7.1.79 DIEPDMA*B*<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device IN Endpoint Buffer Address Register
- **Description:** This register contains the DMA Buffer Address of the IN Endpoint i of the Device controller.  
  
**Note:** This register exists for an endpoint i if the OTG\_EP\_DIR\_i parameter is 0 or 1 for that endpoint.
- **Size:** 32 bits
- **Offset:** 0x91C + i\*20
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 1) && OTG\_EN\_DESC\_DMA==1 && (OTG\_EP\_DIR\_1 !=2)



**Table 7-84 Fields for Register: DIEPDMA*B*<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)**

Bits	Name	Memory Access	Description
31:0	DMABufferAddr	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p> <p><b>Value After Reset:</b> DMA registers are implemented in SPRAM. The reset values of these registers in simulation can be 0/X depending upon the SPRAM model.</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> untestable</p>

### 7.1.80 DOEPCCTL0

- **Name:** Device Control OUT Endpoint 0 Control Register
- **Description:** This register is used to control the characteristics of the OUT Endpoint 0 of the Device controller.
- **Size:** 32 bits
- **Offset:** 0xb00
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6)

31	EPEna
30	EPDis
29:28	RESERVED
27	SNAK
26	CNAK
25:22	RESERVED1
21	Stall
20	Snp
19:18	EPType
17	NAKsts
16	RESERVED2
15	USBActEP
14:2	RESERVED3
1:0	MPS

**Table 7-85 Fields for Register: DOEPCCTL0**

Bits	Name	Memory Access	Description
31	EPEna	R/W1S	<p>Endpoint Enable (EPEna)</p> <ul style="list-style-type: none"> <li>■ 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.</li> <li>■ 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.</li> <li>■ The core clears this bit before setting any of the following interrupts on this endpoint:           <ul style="list-style-type: none"> <li>- SETUP Phase Done</li> <li>- Endpoint Disabled</li> <li>- Transfer Completed</li> </ul> </li> </ul> <p>Note: In DMA mode, this bit must be set for the core to transfer SETUP data packets into memory and it will not be cleared on Transfer Completed interrupt of SETUP packet.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No action</li> <li>■ 0x1 (ACTIVE): Enable Endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-85 Fields for Register: DOEPCCTL0 (Continued)**

Bits	Name	Memory Access	Description
30	EPDis	R	<p>Endpoint Disable (EPDis) The application cannot disable control OUT endpoint 0.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Endpoint disable</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
29:28	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
27	SNAK	W	<p>Set NAK (SNAK) 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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOSET): No action</li> <li>■ 0x1 (SET): Set NAK</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
26	CNAK	W	<p>Clear NAK (CNAK) A write to this bit clears the NAK bit for the endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOCLEAR): No action</li> <li>■ 0x1 (CLEAR): Clear NAK</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
25:22	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-85 Fields for Register: DOEPCCTL0 (Continued)**

Bits	Name	Memory Access	Description
21	Stall	R/W1S	<p>STALL Handshake (Stall)  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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Stall</li> <li>■ 0x1 (ACTIVE): Stall Handshake</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
20	Snp	R/W	<p>RESERVED</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (RESERVED0): Reserved 0</li> <li>■ 0x1 (RESERVED1): Reserved 1</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
19:18	EPType	R	<p>Endpoint Type (EPType)  Hardcoded to 2'b00 for control.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ACTIVE): Endpoint Control 0</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-85 Fields for Register: DOEPCCTL0 (Continued)**

Bits	Name	Memory Access	Description
17	NAKsts	R	<p>NAK Status (NAKsts) Indicates the following:</p> <ul style="list-style-type: none"> <li>■ 1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</li> <li>■ 1'b1: The core is transmitting NAK handshakes on this endpoint.</li> </ul> <p>When either the application or the core sets this bit, the core stops receiving data, even if there is space in the RxFIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): The core is transmitting non-NAK handshakes based on the FIFO status</li> <li>■ 0x1 (ACTIVE): The core is transmitting NAK handshakes on this endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always  <b>Testable:</b> readOnly</p>
16	RESERVED2	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> 0  <b>Testable:</b> writeAsRead  <b>Write Constraint:</b> writeAsRead</p>
15	USBActEP	R	<p>USB Active Endpoint (USBActEP) This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (ACTIVE): USB Active Endpoint 0</li> </ul> <p><b>Value After Reset:</b> 0x1  <b>Exists:</b> Always</p>
14:2	RESERVED3	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> 0  <b>Testable:</b> writeAsRead  <b>Write Constraint:</b> writeAsRead</p>

**Table 7-85 Fields for Register: DOEPCCTL0 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
1:0	MPS	R	<p>Maximum Packet Size (MPS) The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN Endpoint 0.</p> <ul style="list-style-type: none"> <li>■ 2'b00: 64 bytes</li> <li>■ 2'b01: 32 bytes</li> <li>■ 2'b10: 16 bytes</li> <li>■ 2'b11: 8 bytes</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (BYTE64): 64 bytes</li> <li>■ 0x1 (BYTE32): 32 bytes</li> <li>■ 0x2 (BYTE16): 16 bytes</li> <li>■ 0x3 (BYTE8): 8 bytes</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

## 7.1.81 DOEPINT0

- **Name:** Device OUT Endpoint 0 Interrupt Register
  - **Description:** This register contains the interrupts for the OUT Endpoint 0 of the Device controller.
  - **Size:** 32 bits
  - **Offset:** 0xb08
  - **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6)

Rsvd	31:16
StupPktRcvd	15
NYETIntrpt	14
NAKIntrpt	13
BbleErr	12
PktDrpSis	11
Rsvd	10
BNALntr	9
OutPktErr	8
Rsvd	7
Back2BackSETop	6
StsPhseRcvd	5
OUTTknePdis	4
setUp	3
AHBErr	2
EPDisbld	1
XferCompl	0

**Table 7-86 Fields for Register: DOEPINT0**

Bits	Name	Memory Access	Description
31:16			<b>Reserved Field: Yes</b>

**Table 7-86 Fields for Register: DOEPINT0 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
15	StupPktRcvd	R/W1C	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the controller, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the controller closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the controller can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <ul style="list-style-type: none"> <li>■ 1'b0: No Setup packet received</li> <li>■ 1'b1: Setup packet received</li> </ul> <p>Reset: 1'b0</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOT_RCVD): No Setup packet received</li> <li>■ 0x1 (RCVD): Setup packet received</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
14	NYETIntrpt	R/W1C	<p>NYET Interrupt (NYETIntrpt) The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No NYET interrupt</li> <li>■ 0x1 (ACTIVE): NYET Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
13	NAKIntrpt	R/W1C	<p>NAK Interrupt (NAKInterrupt) 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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No NAK interrupt</li> <li>■ 0x1 (ACTIVE): NAK Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-86 Fields for Register: DOEPINT0 (Continued)**

Bits	Name	Memory Access	Description
12	BbleErr	R/W1C	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No BbleErr interrupt</li> <li>■ 0x1 (ACTIVE): BbleErr interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
11	PktDrpSts	R/W1C	<p>Packet Drop Status (PktDrpSts) 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.</p> <p>Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No interrupt</li> <li>■ 0x1 (ACTIVE): Packet Drop Status interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
10			<b>Reserved Field:</b> Yes
9	BNAIntr	R/W1C	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) 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, such as Host busy or DMA done.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No BNA interrupt</li> <li>■ 0x1 (ACTIVE): BNA interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
8	OutPktErr	R/W1C	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No OUT Packet Error</li> <li>■ 0x1 (ACTIVE): OUT Packet Error</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DED_TX_FIFO == 1</p>

**Table 7-86 Fields for Register: DOEPINT0 (Continued)**

Bits	Name	Memory Access	Description
7			<b>Reserved Field:</b> Yes
6	Back2BackSETUp	R/W1C	<p>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. For information about handling this interrupt,</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Back-to-Back SETUP Packets Received</li> <li>■ 0x1 (ACTIVE): Back-to-Back SETUP Packets Received</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ARCHITECTURE !=0</p>
5	StsPhseRcvd	R/W1C	<p>Status Phase Received for Control Write (StsPhseRcvd) This interrupt is valid only for Control OUT endpoints.</p> <p>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.</p> <p>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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Status Phase Received for Control Write</li> <li>■ 0x1 (ACTIVE): Status Phase Received for Control Write</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
4	OUTTknEPdis	R/W1C	<p>OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints.</p> <p>Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No OUT Token Received When Endpoint Disabled</li> <li>■ 0x1 (ACTIVE): OUT Token Received When Endpoint Disabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-86 Fields for Register: DOEPINT0 (Continued)**

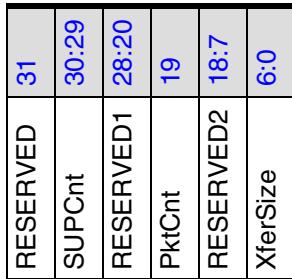
Bits	Name	Memory Access	Description
3	SetUp	R/W1C	<p>SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No SETUP Phase Done</li> <li>■ 0x1 (ACTIVE): SETUP Phase Done</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
2	AHBErr	R/W1C	<p>AHB Error (AHBErr) 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. For details, see "AHB Error Handling" section in the Programming Guide.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No AHB Error Interrupt</li> <li>■ 0x1 (ACTIVE): AHB Error interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
1	EPDisbld	R/W1C	<p>Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Endpoint Disabled Interrupt</li> <li>■ 0x1 (ACTIVE): Endpoint Disabled Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

**Table 7-86 Fields for Register: DOEPINT0 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
0	XferCompl	R/W1C	<p>Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled</p> <ul style="list-style-type: none"> <li>■ For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO.</li> <li>■ 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.</li> </ul> <p>Note: In DMA mode, this bit must be set for the core to transfer SETUP data packets into memory. 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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Transfer Complete Interrupt</li> <li>■ 0x1 (ACTIVE): Transfer Complete Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.82 DOEPTSIZ0

- **Name:** Device OUT Endpoint 0 Transfer Size Register
- **Description:** This register contains the Transfer Size for the OUT Endpoint 0 of the Device controller.
- **Size:** 32 bits
- **Offset:** 0xb10
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6)



**Table 7-87 Fields for Register: DOEPTSIZ0**

Bits	Name	Memory Access	Description
31	RESERVED	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
30:29	SUPCnt	R/W	<p>SETUP Packet Count (SUPCnt)</p> <p>This field specifies the number of back-to-back SETUP data packets the endpoint can receive.</p> <ul style="list-style-type: none"> <li>■ 2'b01: 1 packet</li> <li>■ 2'b10: 2 packets</li> <li>■ 2'b11: 3 packets</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x1 (ONEPACKET): 1 packet</li> <li>■ 0x2 (TWOPACKET): 2 packets</li> <li>■ 0x3 (THREEPACKET): 3 packets</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-87 Fields for Register: DOEPTSIZ0 (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
28:20	RESERVED1	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
19	PktCnt	R/W	<p>Packet Count (PktCnt)</p> <p>This field is decremented to zero after a packet is written into the RxFIFO.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
18:7	RESERVED2	R	<p>RESERVED</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>
6:0	XferSize	R/W	<p>Transfer Size (XferSize)</p> <p>Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only 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.</p> <p>The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.83 DOEPDMA0

- **Name:** Device OUT Endpoint 0 DMA Address Register
- **Description:** This register contains the DMA Address for the OUT Endpoint 0 of the Device controller.
- **Size:** 32 bits
- **Offset:** 0xb14
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6 && OTG\_ARCHITECTURE == 2)



**Table 7-88 Fields for Register: DOEPDMA0**

Bits	Name	Memory Access	Description
31:0	DMAAddr	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>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.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <ul style="list-style-type: none"> <li>■ When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</li> <li>■ When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</li> </ul> <p><b>Value After Reset:</b> DMA registers are implemented in SPRAM. The reset values of these registers in simulation can be 0/X depending upon the SPRAM model.</p> <p><b>Exists:</b> Always</p>

### 7.1.84 DOEPDMAB0

- **Name:** Device OUT Endpoint 0 DMA Buffer Address Register
- **Description:** This register contains the DMA Buffer Address for the OUT Endpoint 0 of the Device controller.
- **Size:** 32 bits
- **Offset:** 0xb1c
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && OTG\_EN\_DESC\_DMA==1



**Table 7-89 Fields for Register: DOEPDMAB0**

Bits	Name	Memory Access	Description
31:0	DMABufferAddr	R	<p>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> <p><b>Value After Reset:</b> DMA registers are implemented in SPRAM. The reset values of these registers in simulation can be 0/X depending upon the SPRAM model.</p> <p><b>Exists:</b> Always  <b>Testable:</b> untestable</p>

### 7.1.85 DOEPCTLi (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device Control OUT Endpoint Control Register
- **Description:** This register is used to control the characteristics of OUT Endpoint i of the Device controller.
- **Size:** 32 bits
- **Offset:** 0xB00 + i\*20
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 1) && (OTG\_EP\_DIR\_1 !=1)

EPEna	31	EPDis	30	SetD1PID	29	SetD0PID	28	SNAK	27	CNAK	26	Rsvd	25:22	Stall	21	Snp	20	EPType	19:18	NAKSts	17	DPID	16	USBActEP	15	Rsvd	14:11	MPS	10:0
-------	----	-------	----	----------	----	----------	----	------	----	------	----	------	-------	-------	----	-----	----	--------	-------	--------	----	------	----	----------	----	------	-------	-----	------

**Table 7-90 Fields for Register: DOEPCCTL<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
31	EPEna	R/W1S	<p>Endpoint Enable (EPEna) Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled,</p> <ul style="list-style-type: none"> <li>■ For IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup.</li> <li>■ For OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup.</li> </ul> <p>When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode:</p> <ul style="list-style-type: none"> <li>■ For IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint.</li> <li>■ For OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB.</li> </ul> <p>The core clears this bit before setting any of the following interrupts on this endpoint:</p> <ul style="list-style-type: none"> <li>■ SETUP Phase Done</li> <li>■ Endpoint Disabled</li> <li>■ Transfer Completed</li> </ul> <p><b>Note:</b> For control endpoints in DMA mode, this bit must be set for the controller to transfer SETUP data packets to the memory. This bit will not be cleared on Transfer Completed interrupt of the SETUP packet.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Action</li> <li>■ 0x1 (ACTIVE): Enable Endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-90 Fields for Register: DOEPCCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
30	EPDis	R/W1S	<p>Endpoint Disable (EPDis) Applies to IN and OUT endpoints.</p> <p>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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Action</li> <li>■ 0x1 (ACTIVE): Disable Endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
29	SetD1PID	W	<p><i>Set DATA1 PID (SetD1PID)</i></p> <ul style="list-style-type: none"> <li>■ Applies to interrupt and bulk IN and OUT endpoints only.</li> <li>■ Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1.</li> <li>■ This field is applicable both for scatter-gather DMA mode and non scatter-gather DMA mode.</li> </ul> <p><b>Reset:</b> 1'b0</p> <p><i>Set Odd (micro)frame (SetOddFr)</i></p> <ul style="list-style-type: none"> <li>■ Applies to isochronous IN and OUT endpoints only.</li> <li>■ Writing to this field sets the even and odd (micro)frame (EO_FrNum) field to odd (micro)frame.</li> </ul> <p><b>Reset:</b> 1'b0</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disables Set DATA1 PID or Do not force Odd Frame</li> <li>■ 0x1 (ENABLED): Set Endpoint Data PID to DATA1 or Sets EO_FrNum field to odd (micro)Frame</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-90 Fields for Register: DOEPCCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
28	SetDOPID	W	<p><i>Set DATA0 PID (SetDOPID)</i></p> <ul style="list-style-type: none"> <li>■ Applies to interrupt/bulk IN and OUT endpoints only.</li> <li>■ Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0.</li> <li>■ This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</li> </ul> <p><b>Reset:</b> 1'b0</p> <p>In non-Scatter/Gather DMA mode: <i>Set Even (micro)frame (SetEvenFr)</i></p> <ul style="list-style-type: none"> <li>■ Applies to isochronous IN and OUT endpoints only.</li> <li>■ Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro)frame.</li> <li>■ 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.</li> </ul> <p><b>Reset:</b> 1'b0</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disables Set DATA0 PID or Do not force Even Frame</li> <li>■ 0x1 (ENABLED): Set Endpoint Data PID to DATA0 or Sets EO_FrNum field to odd (micro)Frame</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
27	SNAK	W	<p><i>Set NAK (SNAK)</i></p> <p>A write to this bit sets the NAK bit for the endpoint.</p> <p>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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Set NAK</li> <li>■ 0x1 (ACTIVE): Set NAK</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-90 Fields for Register: DOEPCCTL<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
26	CNAK	W	<p>Clear NAK (CNAK) A write to this bit clears the NAK bit for the endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Clear NAK</li> <li>■ 0x1 (ACTIVE): Clear NAK</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
25:22			<b>Reserved Field:</b> Yes
21	Stall	R/W1S	<p>STALL Handshake (Stall) 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. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): STALL All non-active tokens</li> <li>■ 0x1 (ACTIVE): STALL All Active Tokens</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
20	Snp	R/W	<p>RESERVED</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (RESERVED0): Reserved 0</li> <li>■ 0x1 (RESERVED1): Reserved 1</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-90 Fields for Register: DOEPCCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
19:18	EPType	R/W	<p>Endpoint Type (EPType)  This is the transfer type supported by this logical endpoint.</p> <ul style="list-style-type: none"> <li>■ 2'b00: Control</li> <li>■ 2'b01: Isochronous</li> <li>■ 2'b10: Bulk</li> <li>■ 2'b11: Interrupt</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (CONTROL): Control</li> <li>■ 0x1 (ISOCHRONOUS): Isochronous</li> <li>■ 0x2 (BULK): Bulk</li> <li>■ 0x3 (INTERRUPT): Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

**Table 7-90 Fields for Register: DOEPCCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
17	NAKSts	R	<p>NAK Status (NAKSts)      Indicates the following:</p> <ul style="list-style-type: none"> <li>■ 1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.</li> <li>■ 1'b1: The core is transmitting NAK handshakes on this endpoint.</li> </ul> <p>When either the application or the core sets this bit:</p> <ul style="list-style-type: none"> <li>■ The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.</li> <li>■ For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.</li> <li>■ For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.</li> </ul> <p>Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NONNAK):          The core is transmitting non-NAK handshakes based on the FIFO status</li> <li>■ 0x1 (NAK):          The core is transmitting NAK handshakes on this endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always  <b>Testable:</b> readOnly</p>

**Table 7-90 Fields for Register: DOEPCCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
16	DPID	R	<p><i>Endpoint Data PID (DPID)</i>      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> <ul style="list-style-type: none"> <li>■ 1'b0: DATA0</li> <li>■ 1'b1: DATA1</li> </ul> <p>This field is applicable for both Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p> <p><b>Reset:</b> 1'b0</p> <p><i>Even/Odd (Micro)Frame (EO_FrNum)</i>      In non-Scatter/Gather DMA mode:</p> <ul style="list-style-type: none"> <li>■ Applies to isochronous IN and OUT endpoints only.</li> <li>■ 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.           <ul style="list-style-type: none"> <li>- 1'b0: Even (micro)frame</li> <li>- 1'b1: Odd (micro)frame</li> </ul> </li> <li>■ 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.</li> </ul> <p><b>Reset:</b> 1'b0</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Endpoint Data PID not active</li> <li>■ 0x1 (ACTIVE): Endpoint Data PID active</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> readOnly</p>

**Table 7-90 Fields for Register: DOEPCCTL*i* (for *i* = 1; *i* <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
15	USBActEP	R/W	<p>USB Active Endpoint (USBActEP)            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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Not Active</li> <li>■ 0x1 (ENABLED): USB Active Endpoint</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>
14:11			<b>Reserved Field:</b> Yes
10:0	MPS	R/W	<p>Maximum Packet Size (MPS)            The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> Always</p>

### 7.1.86 DOEPINT<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device OUT Endpoint Interrupt Register
- **Description:** This register contains the interrupts for the OUT Endpoint i of the Device controller.
- **Size:** 32 bits
- **Offset:** 0xB08 + i\*20
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 1) && (OTG\_EP\_DIR\_1 !=1)

31:16	Rsvd	StupPktRcvd	NYETIntrpt	NAKIntrpt	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				BbleErr				Rsvd	BNAIntr	OutPktErr	Rsvd	Back2BackSETup	StsPhseRcvd	OUTTInEPdis	setUp	AHBErr	EPDisbld	XferCompl

Table 7-91 Fields for Register: DOEPINT<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)

Bits	Name	Memory Access	Description
31:16			<b>Reserved Field:</b> Yes

**Table 7-91 Fields for Register: DOEPINT<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
15	StupPktRcvd	R/W1C	<p>Setup Packet Received Applicable for Control OUT Endpoints in only in the Buffer DMA Mode</p> <p>Set by the controller, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the controller closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address.</p> <p>Note: Because of the above behavior, the controller can receive any number of back to back setup packets and one buffer for every setup packet is used.</p> <ul style="list-style-type: none"> <li>■ 1'b0: No Setup packet received</li> <li>■ 1'b1: Setup packet received</li> </ul> <p>Reset: 1'b0</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOT_RCVD): No Setup packet received</li> <li>■ 0x1 (RCVD): Setup packet received</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
14	NYETIntrpt	R/W1C	<p>NYET Interrupt (NYETIntrpt)</p> <p>The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No NYET interrupt</li> <li>■ 0x1 (ACTIVE): NYET Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
13	NAKIntrpt	R/W1C	<p>NAK Interrupt (NAKInterrupt)</p> <p>The core generates this interrupt when a NAK is transmitted or received by the device.</p> <p>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.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No NAK interrupt</li> <li>■ 0x1 (ACTIVE): NAK Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-91 Fields for Register: DOEPINT<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
12	BbleErr	R/W1C	<p>NAK Interrupt (BbleErr) The core generates this interrupt when babble is received for the endpoint.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No BbleErr interrupt</li> <li>■ 0x1 (ACTIVE): BbleErr interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
11	PktDrpSts	R/W1C	<p>Packet Drop Status (PktDrpSts) 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.</p> <p>Dependency: This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No interrupt</li> <li>■ 0x1 (ACTIVE): Packet Drop Status interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
10			<b>Reserved Field:</b> Yes
9	BNAIntr	R/W1C	<p>BNA (Buffer Not Available) Interrupt (BNAIntr) 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, such as Host busy or DMA done</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No BNA interrupt</li> <li>■ 0x1 (ACTIVE): BNA interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
8	OutPktErr	R/W1C	<p>OUT Packet Error (OutPktErr) Applies to OUT endpoints Only This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No OUT Packet Error</li> <li>■ 0x1 (ACTIVE): OUT Packet Error</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_DED_TX_FIFO == 1</p>

**Table 7-91 Fields for Register: DOEPINT<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
7			<b>Reserved Field:</b> Yes
6	Back2BackSETUp	R/W1C	<p>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. For information about handling this interrupt,</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Back-to-Back SETUP Packets Received</li> <li>■ 0x1 (ACTIVE): Back-to-Back SETUP Packets Received</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
5	StsPhseRcvd	R/W1C	<p>Status Phase Received for Control Write (StsPhseRcvd) This interrupt is valid only for Control OUT endpoints.</p> <p>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.</p> <p>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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Status Phase Received for Control Write</li> <li>■ 0x1 (ACTIVE): Status Phase Received for Control Write</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
4	OUTTknEPdis	R/W1C	<p>OUT Token Received When Endpoint Disabled (OUTTknEPdis) Applies only to control OUT endpoints.</p> <p>Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No OUT Token Received When Endpoint Disabled</li> <li>■ 0x1 (ACTIVE): OUT Token Received When Endpoint Disabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

**Table 7-91 Fields for Register: DOEPINT<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

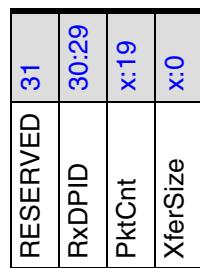
Bits	Name	Memory Access	Description
3	SetUp	R/W1C	<p>SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No SETUP Phase Done</li> <li>■ 0x1 (ACTIVE): SETUP Phase Done</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
2	AHBErr	R/W1C	<p>AHB Error (AHBErr) 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. For details, see "AHB Error Handling" section in the Programming Guide.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No AHB Error Interrupt</li> <li>■ 0x1 (ACTIVE): AHB Error interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>
1	EPDisbld	R/W1C	<p>Endpoint Disabled Interrupt (EPDisbld) Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Endpoint Disabled Interrupt</li> <li>■ 0x1 (ACTIVE): Endpoint Disabled Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0 <b>Exists:</b> Always</p>

**Table 7-91 Fields for Register: DOEPINT<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
0	XferCompl	R/W1C	<p>Transfer Completed Interrupt (XferCompl) Applies to IN and OUT endpoints.</p> <ul style="list-style-type: none"> <li>■ When Scatter/Gather DMA mode is enabled           <ul style="list-style-type: none"> <li>- For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO.</li> <li>- 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.</li> </ul> </li> <li>■ 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.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): No Transfer Complete Interrupt</li> <li>■ 0x1 (ACTIVE): Transfer Complete Interrupt</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.87 DOEPTSIZi (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device OUT Endpoint Transfer Size Register
- **Description:** This register contains the Transfer Size for the OUT Endpoint i of the Device controller.
- **Size:** 32 bits
- **Offset:** 0xB10 + i\*20
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 1) && (OTG\_EP\_DIR\_1 !=1)



**Table 7-92 Fields for Register: DOEPTSIZi (for i = 1; i <= OTG\_NUM\_EPS)**

Bits	Name	Memory Access	Description
31	RESERVED	R	<p><b>RESERVED</b></p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> 0</p> <p><b>Testable:</b> writeAsRead</p> <p><b>Write Constraint:</b> writeAsRead</p>

**Table 7-92 Fields for Register: DOEPTSIZi (for i = 1; i <= OTG\_NUM\_EPS) (Continued)**

Bits	Name	Memory Access	Description
30:29	RxDPID	R	<p>RxDPID</p> <p>Applies to isochronous OUT endpoints only.</p> <p>This is the data PID received in the last packet for this endpoint.</p> <ul style="list-style-type: none"> <li>■ 2'b00: DATA0</li> <li>■ 2'b01: DATA2</li> <li>■ 2'b10: DATA1</li> <li>■ 2'b11: MDATA</li> </ul> <p>SETUP Packet Count (SUPCn)</p> <p>Applies to control OUT Endpoints only.</p> <p>This field specifies the number of back-to-back SETUP data packets the endpoint can receive.</p> <ul style="list-style-type: none"> <li>■ 2'b01: 1 packet</li> <li>■ 2'b10: 2 packets</li> <li>■ 2'b11: 3 packets</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DATA0): DATA0</li> <li>■ 0x1 (DATA2PACKET1): DATA2 or 1 packet</li> <li>■ 0x2 (DATA1PACKET2): DATA1 or 2 packets</li> <li>■ 0x3 (MDATAPACKET3): MDATA or 3 packets</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
x:19	PktCnt	R/W	<p>Packet Count (PktCnt) This field is decremented to zero after a packet is written into the RxFIFO.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Range Variable[x]:</b> OTG_PACKET_COUNT_WIDTH + 18</p>
x:0	XferSize	R/W	<p>Transfer Size (XferSize)</p> <p>Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only 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.</p> <p>The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.</p> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p> <p><b>Range Variable[x]:</b> OTG_TRANS_COUNT_WIDTH - 1</p>

### 7.1.88 DOEPDMA<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device OUT Endpoint DMA Address Register
- **Description:** This register contains the DMA Address for the OUT Endpoint i of the Device controller.
- **Size:** 32 bits
- **Offset:** 0xB14 + i\*20
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && (OTG\_NUM\_EPS >= 1) && (OTG\_ARCHITECTURE == 2) && (OTG\_EP\_DIR\_1 !=1)



**Table 7-93 Fields for Register: DOEPDMA<sub>i</sub> (for i = 1; i <= OTG\_NUM\_EPS)**

Bits	Name	Memory Access	Description
31:0	DMAAddr	R/W	<p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p>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.</p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <ul style="list-style-type: none"> <li>■ When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field.</li> <li>■ When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.</li> </ul> <p><b>Value After Reset:</b> DMA registers are implemented in SPRAM. The reset values of these registers in simulation can be 0/X depending upon the SPRAM model.</p> <p><b>Exists:</b> Always</p>

### 7.1.89 DOEPDMABi (for i = 1; i <= OTG\_NUM\_EPS)

- **Name:** Device OUT Endpoint Buffer Address Register
- **Description:** This register contains the DMA Buffer Address for the OUT Endpoint i of the Device controller.
- **Size:** 32 bits
- **Offset:** 0xB1C + i\*20
- **Exists:** (OTG\_MODE != 5 && OTG\_MODE != 6) && OTG\_EN\_DESC\_DMA==1 && (OTG\_NUM\_EPS >= 1) && (OTG\_EP\_DIR\_1 !=1)



**Table 7-94 Fields for Register: DOEPDMABi (for i = 1; i <= OTG\_NUM\_EPS)**

Bits	Name	Memory Access	Description
31:0	DMABufferAddr	R	<p>Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress.</p> <p>This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.</p> <p><b>Value After Reset:</b> DMA registers are implemented in SPRAM. The reset values of these registers in simulation can be 0/X depending upon the SPRAM model.</p> <p><b>Exists:</b> Always</p> <p><b>Testable:</b> untestable</p>

### 7.1.90 PCGCCTL

- **Name:** Power and Clock Gating Control Register
- **Description:** This register is used to control the Power and Clock Gating characteristics of the controller.
- **Size:** 32 bits
- **Offset:** 0xe00
- **Exists:** Always

RestoreValue	31:14
EssRegRestored	13
ExtndedHibernationSwitch	12
ExtndedHibernationClamp	11
ExtndedHibernationEn	10
RestoreMode	9
ResetAfterSusp	8
L1Suspended	7
PhySleep	6
Enbl_L1Gating	5
Rsvd	4
RstPdwnModule	3
PwrClmp	2
GateHclk	1
StopPclk	0

**Table 7-95 Fields for Register: PCGCCTL**

Bits	Name	Memory Access	Description
31:14	RestoreValue	R/W	<p>Restore Value (RestoreValue) When Hibernation mode is enabled, the RestoreValue needs to be read at SAVE_POINT to store in non-volatile memory and at RESTORE_POINT restored before PCGCCTL.EssRegRestored field is set.</p> <ul style="list-style-type: none"> <li>■ [31] if_dev_mode <ul style="list-style-type: none"> <li>- 1: Device mode, core restored as device</li> <li>- 0: Host mode, core restored as host</li> </ul> </li> <li>■ [30:29] p2hd_prt_spd (PRT speed) <ul style="list-style-type: none"> <li>- 00: HS</li> <li>- 01: FS</li> <li>- 10: LS</li> <li>- 11: Reserved</li> </ul> </li> <li>■ [28:27] p2hd_dev_enum_spd (Device enumerated speed) <ul style="list-style-type: none"> <li>- 00: HS</li> <li>- 01: FS (30/60 MHz clk)</li> <li>- 10: LS</li> <li>- 11: FS (48 MHz clk)</li> </ul> </li> <li>■ [26:20] mac_dev_addr (MAC device address) Device address</li> <li>■ [19] mac_termselect (Termination selection) <ul style="list-style-type: none"> <li>- 0: HS_TERM (Program for High Speed)</li> <li>- 1: FS_TERM (Program for Full Speed)</li> </ul> </li> <li>■ [18:17] mac_xcvrselect (Transceiver select) <ul style="list-style-type: none"> <li>- 00: HS_XCVR (High Speed)</li> <li>- 01: FS_XCVR (Full Speed)</li> <li>- 10: LS_XCVR (Low Speed)</li> <li>- 11: LFS_XCVR (Reserved)</li> </ul> </li> <li>■ [16] sh2pl_prt_ctl[0] <ul style="list-style-type: none"> <li>- 1: prt_power enabled</li> <li>- 0: prt_power disabled</li> </ul> </li> <li>■ [15:14] prt_clk_sel (Refer prt_clk_sel table) Defines port clock select for different speeds.</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_PWR0PT == 2  <b>Testable:</b> untestable</p>

**Table 7-95 Fields for Register: PCGCCTL (Continued)**

Bits	Name	Memory Access	Description
13	EssRegRestored	W	<p>Essential Register Values Restored (EssRegRestored)  A write of one into this field indicates that register values of essential registers have been restored. For definition of essential registers, refer programming flow section for Hibernation.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (NOT_RESTORED): Register values of essential registers are not restored</li> <li>■ 0x1 (RESTORED): Register values of essential registers have been restored</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_PWROPT == 2</p>
12	ExtndedHibernationSwitch	R/W	<p>Extended Hibernation Switch (ExtndedHibernationSwitch)  This signal is used as a VDD switch control signal for.</p> <ul style="list-style-type: none"> <li>■ 1'b0 : the power switch is disabled</li> <li>■ 1'b1 : the power switch is enabled</li> </ul> <p>This bit is available when the configuration option OTG_EN_PWROPT == 3 is selected.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): The Extended Hibernation Power Switch is disabled</li> <li>■ 0x1 (ENABLED): The Extended Hibernation Power Switch is enabled</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_PWROPT == 3</p>
11	ExtndedHibernationClamp	R/W	<p>Extended Hibernation Clamp (ExtndedHibernationClamp)  This signal is used as a clamp / isolation control signal for \${design}_piu hierarchy which remains in the PowerON</p> <ul style="list-style-type: none"> <li>■ 1'b0 : the clamp is disabled</li> <li>■ 1'b1 : the clamp is enabled</li> </ul> <p>This bit is available when the configuration option OTG_EN_PWROPT == 3 is selected.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): The Extended Hibernation clamps are disabled</li> <li>■ 0x1 (ENABLED): The Extended Hibernation clamps are enabled</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_PWROPT == 3</p>

**Table 7-95 Fields for Register: PCGCCTL (Continued)**

Bits	Name	Memory Access	Description
10	ExtndedHibernationEn	R/W	<p>Extended Hibernation Enable (ExtndedHibernationEn) Enable Extended Hibernation This bit is available when the configuration option OTG_EN_PWROPT == 3 is selected.</p> <ul style="list-style-type: none"> <li>■ 1'b0: The Extended Hibernation feature is not enabled. The DWC-otg core will work in normal Hibernation</li> <li>■ 1'b1: The Extended Hibernation feature is enabled</li> </ul> <p>When the external hibernation feature is enabled, the following features are available in the core:</p> <ul style="list-style-type: none"> <li>■ DWC_otg_piupie State Machine inside DWC_otg_piupie becomes active. DWC_otg_mac_piupie is inactive.</li> <li>■ DWC_otg_piupiustup_store inside DWC_otg_piupie becomes active.</li> <li>■ Clamp / ISO logic between DWC_otg_piupie and remaining part of the core is active.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): The Extended Hibernation feature is disabled</li> <li>■ 0x1 (ENABLED): The Extended Hibernation feature is enabled</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_PWROPT == 3</p>

**Table 7-95 Fields for Register: PCGCCTL (Continued)**

<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
9	RestoreMode	R/W	<p>Restore Mode (RestoreMode) The application should program this bit to specify the restore mode during RESTORE POINT before programming PCGCCTL.EssRegRest bit is set.</p> <ul style="list-style-type: none"> <li>■ Host Mode:           <ul style="list-style-type: none"> <li>- 1'b0: Host-initiated Resume, Host-initiated Reset</li> <li>- 1'b1: Device-initiated Remote Wakeup</li> </ul> </li> <li>■ Device Mode:           <ul style="list-style-type: none"> <li>- 1'b0: Device-initiated Remote Wakeup</li> <li>- 1'b1: Host-initiated Resume, Host-initiated Reset</li> </ul> </li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): In Host mode, this bit indicates Host-initiated Resume and Reset. In Device mode, this bit indicates Device-initiated Remote Wakeup</li> <li>■ 0x1 (ENABLED): In Host mode, this bit indicates Device-initiated Remote Wakeup. In Device mode, this bit indicates Host-initiated Resume and Reset</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_PWRROPT == 2</p>
8	ResetAfterSusp	R/W	<p>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> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): In Host-only mode, host issues Resume after Suspend</li> <li>■ 0x1 (ENABLED): In Host-only mode, host sets this bit before clamp is removed if the host needs to issue Reset after Suspend</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_PWRROPT == 1 &amp;&amp; (OTG_MODE != 3 &amp;&amp; OTG_MODE != 4)</p>

**Table 7-95 Fields for Register: PCGCCTL (Continued)**

Bits	Name	Memory Access	Description
7	L1Suspended	R	<p>L1 Deep Sleep Indicates that the PHY is in deep sleep when in L1 state.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Non Deep Sleep</li> <li>■ 0x1 (ACTIVE): Deep Sleep</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
6	PhySleep	R	<p>PHY In Sleep Indicates that the PHY is in Sleep State.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (INACTIVE): Phy not in Sleep state</li> <li>■ 0x1 (ACTIVE): Phy in Sleep state</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
5	Enbl_L1Gating	R/W	<p>Enable Sleep Clock Gating If this bit is set, core internal clock gating is enabled sleep state if utmi_l1_suspend_n cannot be asserted by the core. The PHY clock will not be gated in sleep state if Enbl_L1Gating is not set.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): The PHY clock is not gated in Sleep state</li> <li>■ 0x1 (ENABLED): The Core internal clock gating is enabled in Sleep state</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_ENABLE_LPM ==1</p>
4			<b>Reserved Field:</b> Yes

**Table 7-95 Fields for Register: PCGCCTL (Continued)**

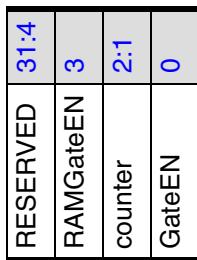
<b>Bits</b>	<b>Name</b>	<b>Memory Access</b>	<b>Description</b>
3	RstPdwnModule	R/W	<p>Reset Power-Down Modules (RstPdwnModule) This bit is valid only in Partial Power-Down mode.</p> <ul style="list-style-type: none"> <li>■ The application sets this bit when the power is turned off.</li> <li>■ The application clears this bit after the power is turned on and the PHY clock is up.</li> </ul> <p>Note: The R/W of all core registers are possible only when this bit is set to 1b0.</p> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (ON): Power is turned on</li> <li>■ 0x1 (OFF): Power is turned off</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>
2	PwrClmp	R/W	<p>Power Clamp (PwrClmp) This bit is valid only in Partial Power-Down mode</p> <ul style="list-style-type: none"> <li>■ 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.</li> <li>■ The application clears the bit to disable the clamping before the power is turned on.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Clears this bit to disable the clamping before the power is turned on</li> <li>■ 0x1 (ENABLED): In only Partial Power-Down mode, sets this bit to clamp the signals between the power-on modules and the power-off modules before the power is turned off</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_PWROPT == 1</p>

**Table 7-95 Fields for Register: PCGCCTL (Continued)**

Bits	Name	Memory Access	Description
1	GateHclk	R/W	<p>Gate Hclk (GateHclk)</p> <ul style="list-style-type: none"> <li>■ 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.</li> <li>■ The application clears this bit when the USB is resumed or a new session starts.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Clears this bit when the USB is resumed or a new session starts</li> <li>■ 0x1 (ENABLED): Sets this bit to gate hclk to modules when the USB is suspended or the session is not valid</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> OTG_EN_PWR0PT == 1</p>
0	StopPclk	R/W	<p>Stop Pclk (StopPclk)</p> <ul style="list-style-type: none"> <li>■ 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.</li> <li>■ The application clears this bit when the USB is resumed or a new session starts.</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): Disable Stop Pclk</li> <li>■ 0x1 (ENABLED): Enable Stop Pclk</li> </ul> <p><b>Value After Reset:</b> 0x0</p> <p><b>Exists:</b> Always</p>

### 7.1.91 PCGCCTL1

- **Name:** Power and Clock Gating Control Register1
- **Description:** This register is used to control the Power and Clock Gating characteristics of the controller.
- **Size:** 32 bits
- **Offset:** 0xe04
- **Exists:** (OTG\_EN\_ACG==1 && OTG\_MODE !=0)



**Table 7-96 Fields for Register: PCGCCTL1**

Bits	Name	Memory Access	Description
31:4	RESERVED	R	<b>RESERVED</b> <b>Value After Reset:</b> 0x0 <b>Exists:</b> Always <b>Testable:</b> untestable
3	RAMGateEN	R/W	RAM Clock Gating Enable (RAMGateEn) The application programs RAMGateEn to enable RAM Clock Gating in the ACG feature. <ul style="list-style-type: none"> <li>■ 1b0: Disable RAM Clock Gating (Default)</li> <li>■ 1b1: Enable RAM Clock Gating</li> </ul> <b>Values:</b> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): RAM Clock Gating disable</li> <li>■ 0x1 (ENABLED): RAM Clock Gating enable</li> </ul> <b>Value After Reset:</b> 0x0 <b>Exists:</b> OTG_EN_ACG==1

**Table 7-96 Fields for Register: PCGCCTL1 (Continued)**

Bits	Name	Memory Access	Description
2:1	counter	R/W	<p>Count to Gate Clock (CntGateClk)      CntGateClk indicates to the Controller how many PHY Clock cycles and AHB Clock cycles of 'IDLE' (no activity) the Controller will wait for before Gating the respective PHY and AHB clocks internal to the Controller.</p> <ul style="list-style-type: none"> <li>■ 2b00: 64 clocks (Default).</li> <li>■ 2b01: 128 clocks</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (CLOCKS64): CLOCKS64</li> <li>■ 0x1 (CLOCKS128): CLOCKS128</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_ACG==1</p>
0	GateEN	R/W	<p>Enable Active Clock Gating (GateEn)      The application programs GateEn to enable Active Clock Gating feature for the PHY and AHB clocks.</p> <ul style="list-style-type: none"> <li>■ 1b0: Disable Active Clock Gating (Default)</li> <li>■ 1b1: Enable Active Clock Gating</li> </ul> <p><b>Values:</b></p> <ul style="list-style-type: none"> <li>■ 0x0 (DISABLED): ACG disable</li> <li>■ 0x1 (ENABLED): ACG enable</li> </ul> <p><b>Value After Reset:</b> 0x0  <b>Exists:</b> OTG_EN_ACG==1</p>



# 8

## Unified Power Format Support

---

This chapter describes how DWC\_otg controller supports the IEEE 1801 standard. It contains the following sections:

- “Overview”
- “Power Clamps”
- “Power Switch Implementation Based on Polarity”
- “UPF Support for the DWC\_otg Controller”
  - “Requirements for UPF-based Power Aware Verification and Synthesis”
  - “UPF-Related Files in the HS OTG Controller” on page 689
- “DWC\_otg Hibernation Power Domains” on page 690
- “DWC\_otg Extended Hibernation Power Domains” on page 691



**Note** For information on running power-aware simulations and synthesis, and using UPF with different simulators, refer to the *DesignWare Cores USB 2.0 Hi-Speed On-The-Go (OTG) User Guide*.

---

### 8.1 Overview

Unified Power Format (UPF) provides the ability for electronic systems to be designed with power as a key consideration early in the process. It enables you to specify implementation-relevant power information early in the design process. UPF also provides a consistent format to specify power-aware design information that cannot be specified in HDL (Hardware Description Language) code or when it is undesirable to directly specify within the HDL logic, as doing so would tie the logic specification directly to a constrained power implementation. UPF also defines consistent semantics across verification and implementation ensuring that what is implemented in the design is the same as what has been verified.

## 8.2 Power Clamps

In the DWC\_otg controller, insertion of clamp cells between the always ON power domain and the power OFF domain can be implemented in the following ways:

- RTL-based power clamps: In this method, the clamp cells are directly instantiated in the RTL.
- UPF-based power clamps: In this method, the clamp cells are instantiated using the UPF file.

### Enabling UPF Power Clamps:

- If you select “Yes” for the “Enable UPF Power Clamps?” configuration parameter in coreConsultant, UPF-based clamps are supported.
- <sup>1</sup>If you select “No” for this parameter, then RTL-based clamps are instantiated.

### Dependency on Power Optimization Option:

- When the Extended Hibernation feature is enabled (OTG\_EN\_PWROPT=3), UPF-based power clamps are not supported. You can use only RTL-based power clamps.
- When the Partial Power Down or Hibernation feature is enabled (OTG\_EN\_PWROPT=1, or 2), both RTL-based and UPF-based power clamps are supported.

## 8.3 Power Switch Implementation Based on Polarity

In the DWC\_otg controller, insertion of polarity-based power switch is supported. You can select the switch polarity using the OTG\_PWR\_SWITCH\_POLARITY coreConsultant parameter.

- When “Active Low” is selected,
  - the switch will be ON if the control signal to the switch is low
  - the switch will be OFF if the control signal to the switch is high
- When “Active High” is selected,
  - the switch will be ON if the control signal to the switch is high
  - the switch will be OFF if the control signal to the switch is low

1. This option is no longer supported.

## 8.4 UPF Support for the DWC\_otg Controller

The HS OTG controller supports the UPF feature for verification with MVTOOLS and Synthesis with DesignCompiler only when the coreConsultant parameter OTG\_EN\_PWR0PT = 1 | 2 | 3 (Partial power-down, Hibernation, or Extended Hibernation is selected).

### 8.4.1 Requirements for UPF-based Power Aware Verification and Synthesis

The following are the requirements to support the UPF flow for the HS OTG controller:

- VCS NLP license for multi voltage simulations
- Synthesis tools with UPF support
- Technology libraries supporting UPF instantiated cells such as isolation cells, level shifters, power switches, and so on

### 8.4.2 UPF-Related Files in the HS OTG Controller

The UPF files for the HS OTG controller are located at <install\_dir>/pkg/powerIntent/DWC\_otg.upf. [Table 8-1](#) lists and describes the various UPF files.

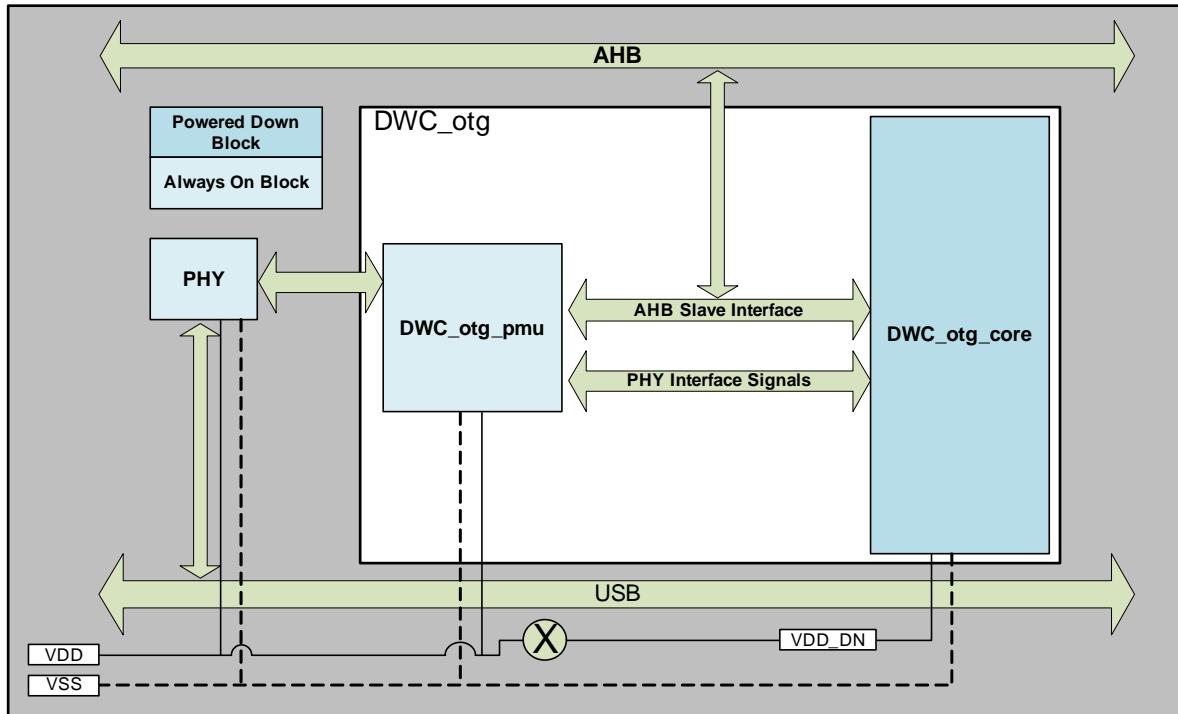
**Table 8-1 UPF Flow-Related Files**

File	Description
README	UPF flow related information and setup instructions.
DWC_otg.upf	UPF file for Spyglass, Simulation, Synthesis and VC LP flows.
set_voltage.tcl	Physical constraints file to set the voltage values

## 8.5 DWC\_otg Hibernation Power Domains

As shown in [Figure 8-1](#), the hibernation feature along with UPF allows the DWC\_otg\_core module to be completely power gated to conserve power. The DWC\_otg\_pmu block remains switched on.

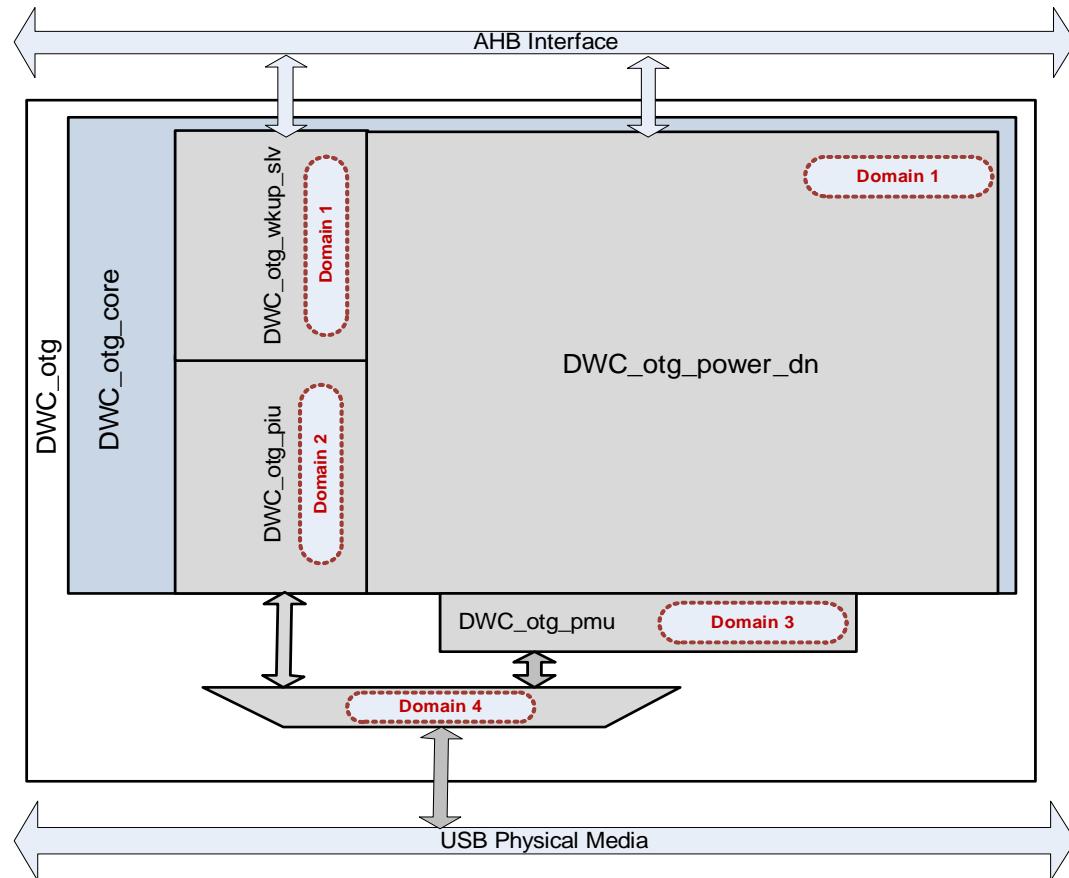
**Figure 8-1** **DWC\_otg Power Domains**



## 8.6 DWC\_otg Extended Hibernation Power Domains

Figure 8-2 shows the DWC\_otg extended hibernation power domains when OTG\_EN\_PWRLOPT == 3.

**Figure 8-2** **DWC\_otg Power Domains with OTG\_EN\_PWRLOPT == 3**



The PHY is always in the PowerON domain. The following power domains are present:

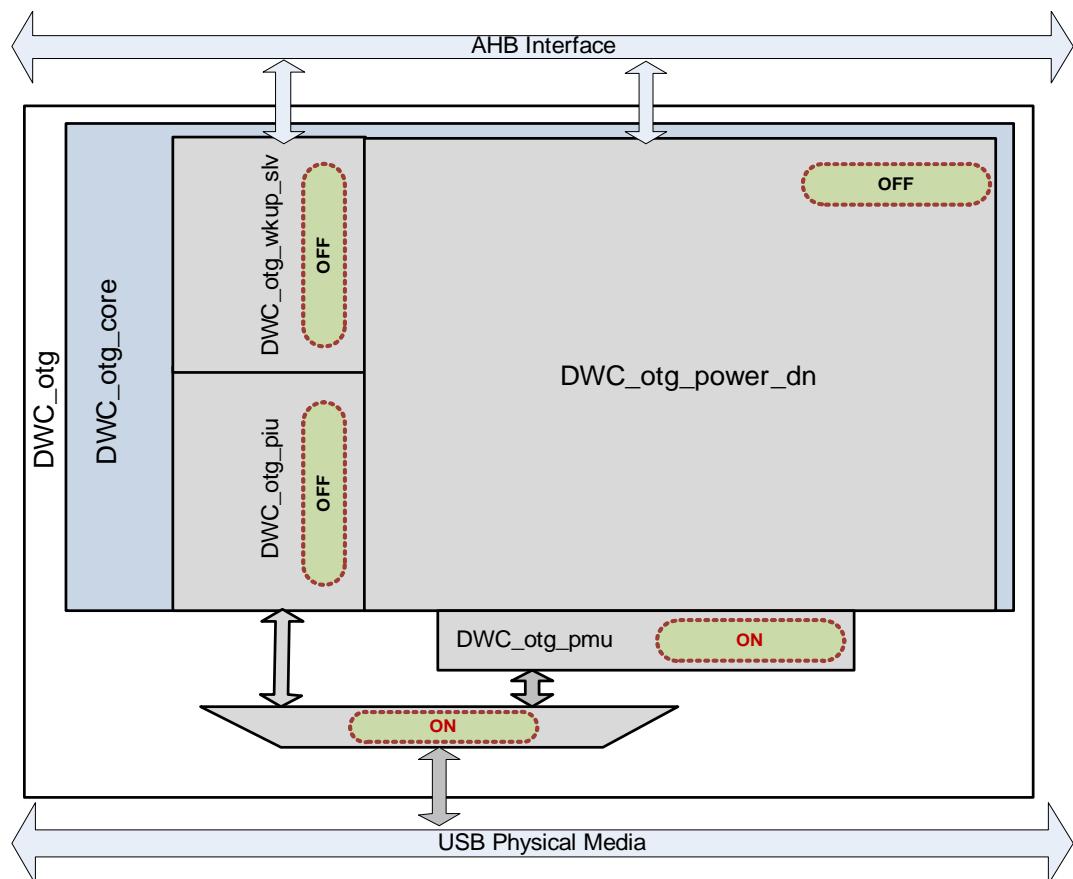
- Power Domain 1: This domain consists of all modules inside the **DWC\_otg\_core** hierarchy, except the **DWC\_otg\_piup** module.
- Power Domain 3: This domain consists of the **DWC\_otg\_pmu** module.
- Power Domain 4: This domain consists of the **DWC\_otg\_xhib\_mux** module.

These power domains provide the flexibility to choose between normal and extended hibernation based on your functional requirement. The following table indicates the domains and their status for different functionalities.

Functionality	Power Domain 1	Power Domain 2	Power Domain 3	Power Domain 4
Normal Operation	PowerON	PowerON	PowerON	PowerON
Hibernation	PowerOff	PowerOff	PowerON	PowerON
Extended Hibernation	PowerOff	PowerON	PowerOFF	PowerON

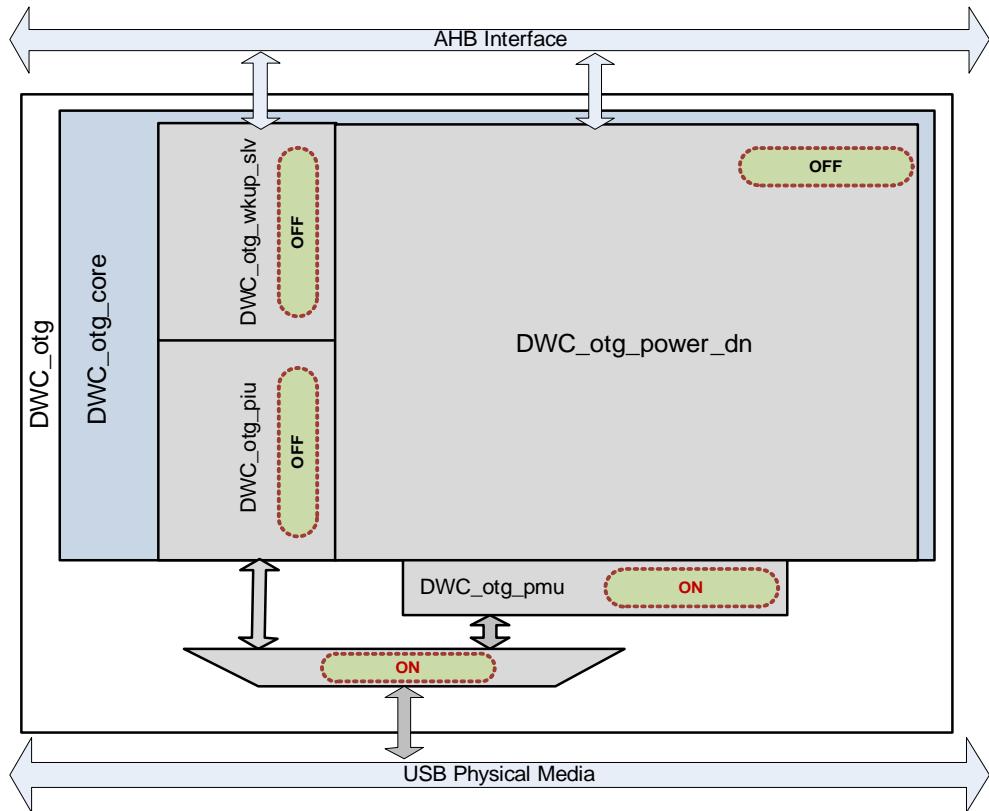
Figure 8-3 shows the ON/OFF power domains of the DWC\_otg during Extended hibernation when OTG\_EN\_PWRLOPT == 3.

**Figure 8-3 DWC\_otg ON/OFF Power Domains During Extended Hibernation with OTG\_EN\_PWRLOPT == 3**



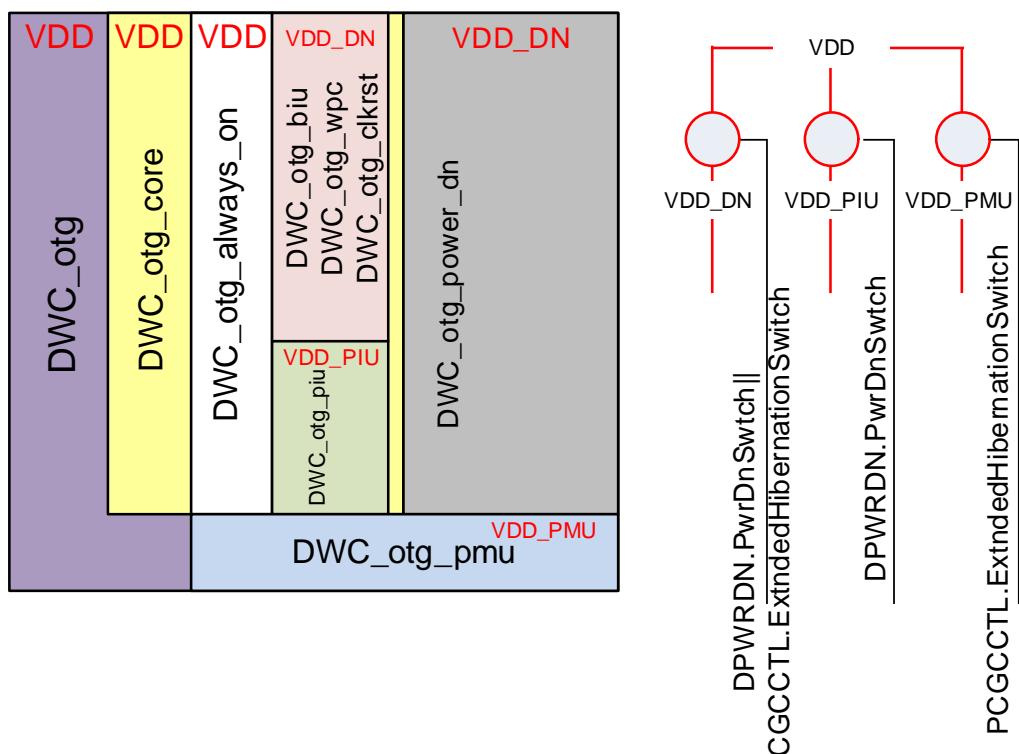
[Figure 8-4](#) shows the ON/OFF power domains of the DWC\_otg during hibernation when OTG\_EN\_PWR0PT == 3.

**Figure 8-4 DWC\_otg ON/OFF Power Domains During Hibernation with OTG\_EN\_PWR0PT == 3**



The PHY is always in the PowerON domain. If normal hibernation functionality is not required, you can combine the Power Domain 1 and Power Domain 2 to form one power domain.

[Figure 8-5](#) on page 694 indicates the VDD connectivity of different modules in the DWC\_otg for External Hibernation feature.

**Figure 8-5 Power Domain Hierarchy for Extended Hibernation Feature**

# A

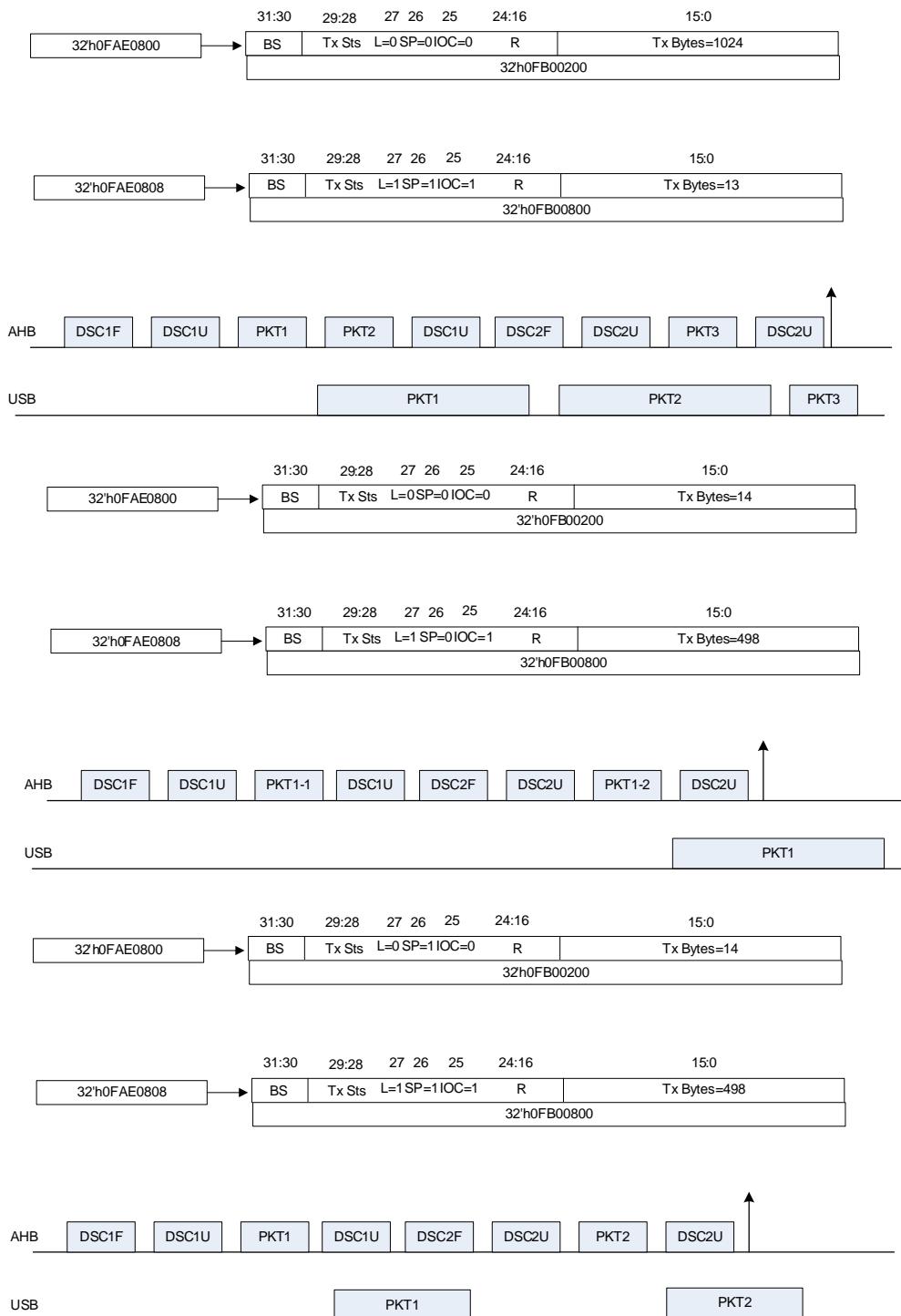
## Example Transfers in Device Scatter/Gather DMA Mode

---

This appendix demonstrates Scatter/Gather DMA functionality.

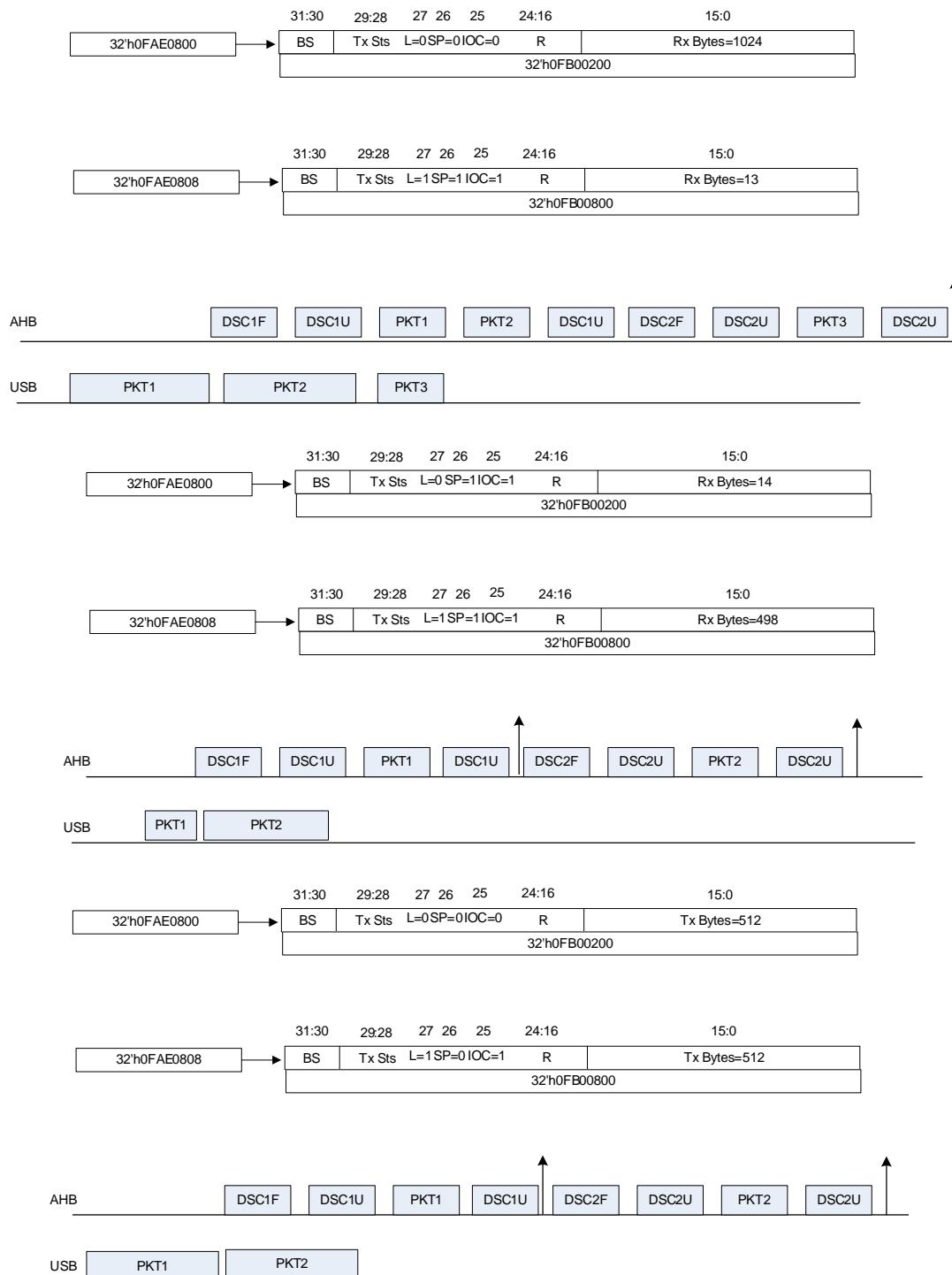
## A.1 Example Transfer: Bulk IN

**Figure A-1 Bulk IN Transfer**



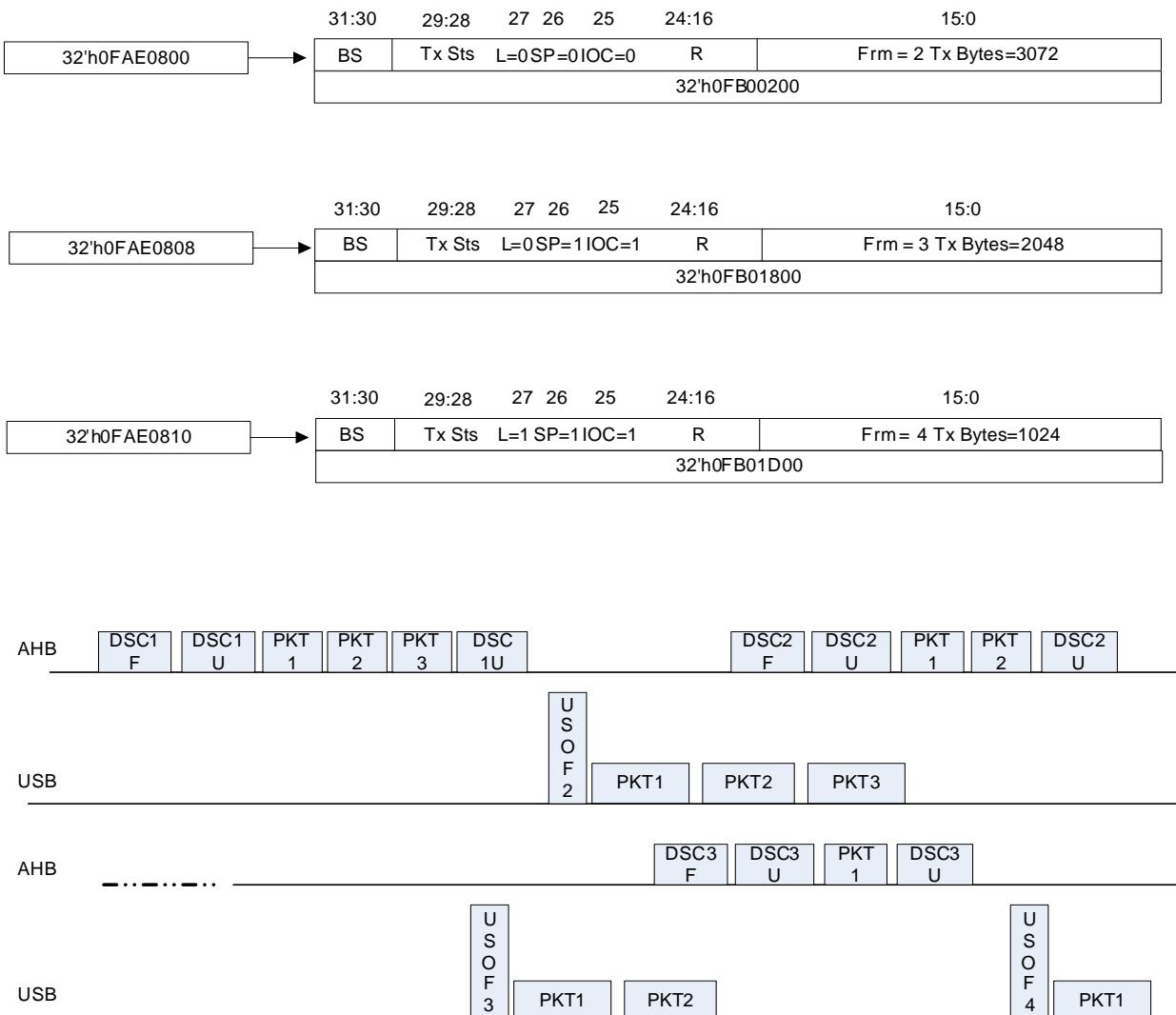
## A.2 Example Transfer: Bulk OUT

**Figure A-2 Bulk OUT Transfer**



## A.3 Example Transfer: Isochronous IN

**Figure A-3 Isochronous IN**



# B

## Area, Speed, and Power

---

This appendix describes the area, speed, and power requirements for several example configurations of the DWC\_otg controller.

### B.1 Area

Tables B-1 to B-8 show the areas for different hardware architecture parameters with different numbers of bidirectional device endpoints/host channels.



The following are the examples and representative of the configuration used. These will vary between controller releases. For a specific configuration on a specific version of the controller, use the Synthesis Activity in coreConsultant to get the exact Area, Speed and Power matching your target technology library:

- The area numbers are in Kilo-gates (K = 1,000 gates) for a well known 0.065 µm technology. The total cell area is divided by the area of the smallest NAND cell in the library to get the area numbers. In general, the gate count difference between 0.065 µm and 0.040 µm is approximately 1k to 2k. Tables B-1 and B-2 illustrate the difference in gate count for shared FIFO mode of operations. All other tables show the gate count when 0.065 µm libraries are used.
  - The Periodic Transfer Interrupt enhancement from Release 2.72a adds 600 gates if a (non-scatter-gather) Buffer DMA configuration is selected.
  - There is an approximate gate count increase of 5K due to Host Scatter/Gather DMA functionality compared to the corresponding Device Scatter/Gather DMA functionality.
-

Tables B-1 and B-2 show areas for different OTG (host and device) configurations for Shared FIFO operation using 65lp and 40lp libraries respectively. Because Device mode endpoint registers are reused as Host mode channel registers, Synopsys recommends setting number of host channels equal to the number of unidirectional endpoints.

**Table B-1 OTG Mode Area for Different Architectures vs. Device Endpoints/Host Channels (K Gates): Shared FIFO Operation (run with 65lp Libraries)**

**Main Parameters:** OTG\_MODE = 0, OTG\_DFIIFO\_DEPTH = 1024, OTG\_HSPHY\_INTERFACE = 1, = 2, OTG\_NPERIO\_TX\_QUEUE\_DEPTH = 8, OTG\_PERIO\_TX\_QUEUE\_DEPTH = 8, OTG\_TOKEN\_QUEUE\_DEPTH = 0, OTG\_TRANS\_COUNT\_WIDTH = 11, OTG\_PACKET\_COUNT\_WIDTH = 4, OTG\_RM\_OPT\_FEATURES = 0, OTG\_MIN\_AHB\_FREQ\_LESS\_THAN\_60 = 1, \*\_FILTER = 0 OTG\_EN\_DESC\_DMA=0

Synthesis AHB clock = 100 MHz, PHY Clock = 60 MHz

BiDir EPs/Host Channels	1/2	2/4	3/6	4/8	5/10	6/12	7/14	8/16	9/16	10/16	11/16	12/16	13/16	14/16	15/16
OTG-Slave-SinglePoint-NoDynamicFifo	34.37	37.29	39.8	42.91	45.64	47.88	50.29	54.1	55.77	57.42	58.77	60.37	61.76	63.38	65.15
OTG-Slave-MultiPoint-NoDynamicFifo	35.02	38.33	41.29	44.68	47.79	50.53	53.32	57.51	59.36	60.77	62.27	63.58	65.24	66.77	68.42
OTG-Slv-MultiPt-DynamicFifo	35.76	39.1	41.99	45.39	48.49	51.26	54	58.23	59.92	61.57	62.96	64.41	65.95	67.52	69.25
OTG-ExtDMA-MultiPt-DynamicFifo	38.13	41.39	44.58	47.84	51.19	54.17	57.3	60.84	62.65	64.41	65.72	67.31	68.96	70.26	72.11
OTG-IntDMA-MultiPt-DynamicFifo	44.3	47.82	51.04	54.59	57.98	61.29	64.38	68.21	70.22	71.84	73.34	74.98	76.82	78.31	80.21
OTG-Slv-MultiPt-DynamicFifo + Tx-Periodic Support	36.47	40.43	43.99	47.91	51.65	54.97	58.38	63.2	65.56	67.69	69.67	71.7	73.85	76.05	78.38
OTG-IntDMA-MultiPt-DynamicFifo + Tx-Periodic Support	45.1	49.15	52.98	57.12	61.36	65.17	68.88	73.09	75.84	78.04	80.39	82.4	84.7	86.87	89.3

**Table B-2 OTG Mode Area for Different Architectures vs. Device Endpoints/Host Channels (K Gates): Shared FIFO Operation (run with 40lp Libraries)**

**Main Parameters:** OTG\_MODE = 0, OTG\_DFIIFO\_DEPTH = 1024, OTG\_HSPHY\_INTERFACE = 1, = 2, OTG\_NPERIO\_TX\_QUEUE\_DEPTH = 8, OTG\_PERIO\_TX\_QUEUE\_DEPTH = 8, OTG\_TOKEN\_QUEUE\_DEPTH = 0, OTG\_TRANS\_COUNT\_WIDTH = 11, OTG\_PACKET\_COUNT\_WIDTH = 4, OTG\_RM\_OPT\_FEATURES = 0, OTG\_MIN\_AHB\_FREQ\_LESS\_THAN\_60 = 1, \*\_FILTER = 0 OTG\_EN\_DESC\_DMA=0

Synthesis AHB clock = 100 MHz, PHY Clock = 60 MHz

BiDir EPs/Host Channels	1/2	2/4	3/6	4/8	5/10	6/12	7/14	8/16	9/16	10/16	11/16	12/16	13/16	14/16	15/16
OTG-Slave-SinglePoint-NoDynamicFifo	35.98	39.09	41.71	44.93	47.87	50.3	52.83	56.52	58.52	60.2	61.74	63.37	64.98	66.6	68.48
OTG-Slave-MultiPoint-NoDynamicFifo	36.68	40.15	43.21	46.84	50.07	53.19	55.99	60.13	62.15	63.95	65.49	67.01	68.67	70.31	72.1
OTG-Slv-MultiPt-DynamicFifo	37.5	40.98	43.95	47.61	50.87	53.88	56.88	60.89	62.95	64.54	66.24	67.8	69.4	71.03	72.81
OTG-ExtDMA-MultiPt-DynamicFifo	39.92	43.39	46.74	50.25	53.85	57.05	60.3	63.84	65.94	67.71	69.26	70.8	72.47	74.01	76.28
OTG-IntDMA-MultiPt-DynamicFifo	46.56	50.38	53.93	57.46	61.13	64.6	67.83	71.87	73.91	75.95	77.49	79.19	81.07	82.85	84.49
OTG-Slv-MultiPt-DynamicFifo + Tx-Periodic Support	38.22	42.35	46.06	50.27	54.1	57.83	61.34	66.06	68.88	71.05	73.33	75.48	77.76	79.96	82.48
OTG-IntDMA-MultiPt-DynamicFifo + Tx-Periodic Support	47.32	51.75	55.94	60.3	64.41	68.57	72.56	76.95	80.06	82.44	84.82	87.33	89.74	92.25	94.58

[Table B-3](#) shows, for Shared FIFO operations, the area required to add endpoint for different DWC\_otg configurations.



[Table B-3](#) values (Additional Area of 1 BiDirEP + 2 Host Channels, Additional Area of 1 BiDirEP) are derived from [Table B-1](#).

**Table B-3 OTG Mode Area Per Configuration Per Endpoint**

Configuration	Additional Area of One Bidirectional Endpoint and Two Host Channels	Additional Area of One Bidirectional Endpoint
OTG-Slave-SinglePoint-NoDynamicFifo	2.82K	1.58K
OTG-Slave-MultiPoint-NoDynamicFifo	3.21K	1.56K
OTG-Slv-MultiPt-DynamicFifo	3.21K	1.57K
OTG-ExtDMA-MultiPt-DynamicFifo	3.24K	1.61K
OTG-IntDMA-MultiPt-DynamicFifo	3.42K	1.71K
OTG-Slv-MultiPt-DynamicFifo + TxISOC	3.82K	2.17K
OTG-IntDMA-MultiPt-DynaFifo + TxISOC	4K	2.32K

[Table B-4](#) shows areas for different Device-only configurations for Shared FIFO operations.

**Table B-4 Device Mode Area for Different Architectures vs. Device Endpoints (K Gates): Shared FIFO Operation**

**Main Parameters:** OTG\_MODE = 4, OTG\_DFIFO\_DEPTH = 1024, OTG\_HSPHY\_INTERFACE = 1, = 2, OTG\_NPERIO\_TX\_QUEUE\_DEPTH = 8, OTG\_PERIO\_TX\_QUEUE\_DEPTH = 8, OTG\_TOKEN\_QUEUE\_DEPTH = 0, OTG\_TRANS\_COUNT\_WIDTH = 11, OTG\_PACKET\_COUNT\_WIDTH = 4, OTG\_RM\_OPT\_FEATURES = 0, OTG\_MIN\_AHB\_FREQ\_LESS\_THAN\_60 = 1, \*\_FILTER = 0 OTG\_EN\_DESC\_DMA=0

Synthesis AHB clock = 100 MHz, PHY Clock = 60 MHz

BiDir EPs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Device-Slave NoDynamicFifo	24.65	25.11	27.06	28.8	30.73	32.39	34.14	36.14	38	39.77	41.02	42.72	44.37	46.03	47.69
Device-Slave DynamicFifo	25.3	25.77	27.61	29.45	31.41	33.03	34.77	36.71	38.69	40.33	41.68	43.37	45.09	46.82	48.32
Device-ExtDMA-DynamicFifo	26.27	26.82	28.75	30.67	32.76	34.48	36.36	38.54	40.4	42.21	43.81	45.46	47.21	48.92	50.66
Device-IntDMA-DynamicFifo	31.69	32.24	34.17	36.16	38.39	40.27	42.02	44.3	46.32	48.11	49.69	51.55	53.46	55.32	56.92
Device-Slave DynamicFifo + Tx-Periodic Support	26.01	27.06	29.52	31.94	34.49	36.69	39.02	41.44	44.28	46.43	48.56	50.67	53	55.21	57.42
Device-IntDMA DynamicFifo + Tx-Periodic	32.34	33.53	36.04	38.7	41.45	43.89	46.35	49.14	51.99	54.28	56.46	58.85	61.45	63.71	65.93

[Table B-5](#) shows the area required to add endpoints for different DWC\_otg configurations when configuration parameter OTG\_MODE = 4 (Non-SRP-Capable Device).

Note that [Table B-5](#) values (Area of 1 BiDirEP) is derived from [Table B-4](#).

**Table B-5 Device Mode Area per Configuration per Endpoint: Shared FIFO Operation**

Configuration	Additional Area of One Bidirectional Endpoint
Device-Slave-NoDynamicFifo	1.65K
Device-Slave-DynamicFifo	1.64K
Device-ExtDMA-DynamicFifo	1.74K
Device-IntDMA-DynamicFifo	1.8K
Device-Slv-DynamicFifo+Tx-Periodic Support	2.24K
Device-IntDMA-DynamicFifo+Tx-Periodic Support	2.4K

[Table B-6](#) shows areas for different Host-only configurations.

**Table B-6 Host Mode Area for Different Architectures vs. Host Channels (K Gates)**

**Main Parameters:** OTG\_MODE = 6, OTG\_DFILO\_DEPTH = 1024, OTG\_HSPHY\_INTERFACE = 1, = 2, OTG\_NPERIO\_TX\_QUEUE\_DEPTH = 8, OTG\_PERIO\_TX\_QUEUE\_DEPTH = 8, OTG\_TOKEN\_QUEUE\_DEPTH = 0, OTG\_TRANS\_COUNT\_WIDTH = 11, OTG\_PACKET\_COUNT\_WIDTH = 4, OTG\_RM\_OPT\_FEATURES = 0, OTG\_MIN\_AHB\_FREQ\_LESS\_THAN\_60 = 1, \*\_FILTER = 0

Synthesis AHB clock = 100 MHz, PHY Clock = 60 MHz

Host Channels	2	4	6	8	10	12	14	16
Host-Slave-SinglePoint-NoDynamicFifo	24.57	25.2	27.11	28.97	30.99	32.8	34.67	36.67
Host-Slave-MultiPoint-NoDynamicFifo	25.24	26.27	28.58	30.82	33.28	35.48	37.74	40.16
Host-Slv-MultiPt-DynamicFifo	26	27.01	29.33	31.58	34.01	36.23	38.49	40.92
Host-ExtDMA-MultiPt-DynamicFifo	28.14	29.34	31.87	34.36	37.02	39.43	41.91	44.55
Host-IntDMA-MultiPt-DynamicFifo	34.14	35.28	37.98	40.56	43.32	45.89	48.46	51.2
Host-Slv-MultiPt-DynamicFifo + Tx-Periodic Support	27.43	28.41	30.71	32.95	35.45	37.63	39.91	42.32
Host-IntDMA-MultiPt-DynamicFifo + Tx-Periodic Support	35.53	36.72	39.43	41.96	44.8	47.31	49.88	52.71

[Table B-7](#) shows the area required to add host-channel for different DWC\_otg configurations when configuration parameter OTG\_MODE = 6 (Non-SRP-Capable Host).

Note that [Table B-7](#) values (Additional Area of 2 Host Channels) are derived from [Table B-6](#). [Table B-8](#) shows the areas required for DWC\_otg features other than endpoints.

**Table B-7 Area per Configuration per Host Channel**

Configuration	Additional Area of Two Host Channels
Host-Slave-SinglePoint-NoDynamicFifo	1.73K
Host-Slave-MultiPoint-NoDynamicFifo	2.13K
Host-Slv-MultiPt-DynamicFifo	2.13K
Host-ExtDMA-MultiPt-DynamicFifo	2.34K
Host-IntDMA-MultiPt-DynamicFifo	2.44K
Host-Slv-MultiPt-DynamicFifo + TxISOC	2.13K
Host-IntDMA-MultiPt-DynaFifo + TxISOC	2.45K

**Table B-8 Areas for DWC\_otg Features**

Feature	Area
ULPI Wrapper	1.65K
Dynamic 8/16 Bit PHY Interface	0.85K
Vendor Control Interface	0.31K
Optional Features (GPIO, User ID register, SOF counter, update toggle)	0.62K
2-deep sink/src buffer (when min AHB frequency < 60MHz)	1.42K
iddig, vbus_valid a_valid, b_valid, and session_end Filters	0.84K
Increase in BiDir EP area for each additional bit of Transfer or Packet Counter (from 11/4 to 19/10 bits)	0.06K
Increase in BiDir EP area for each additional address bit of the FIFO RAM (from 256–32,768 deep)	0.03K
Dynamic HNP/SRP selection (OTG_MODE 0 vs. 2)	3.22K
SRP Support (OTG_MODE 1 vs. 2)	2.9K
USB 1.1 Full-Speed Serial Transceiver Interface	3.33K
I <sup>2</sup> C Interface	1.47K
ULPI Carkit feature	80 gates
Host Periodic Queue /Non-Periodic queue 9 8 deep)	0.44K
Learning Queue	0.07K per entry
Pwer Down Mode (OTG_EN_PWROPT=1)	1.34K
Multi Processor Interrupt (MPI)	200 gates per EP direction

**Table B-9 OTG Mode Area for Various Configurations: Dedicated FIFO**

**Main Parameters:** OTG\_MODE=0, OTG\_TX\_DINEP\_DFIFO\_DEPTH\_n=1024, OTG\_HSPHY\_INTERFACE=1, OTG\_FSPHY\_INTERFACE=0 =1, OTG\_TRANS\_COUNT\_WIDTH=16 OTG\_PACKET\_COUNT\_WIDTH=4, OTG\_RX\_DFIFO\_DEPTH=256 OTG\_TX\_NPERIO\_DFIFO\_DEPTH=256 OTG\_TX\_HNPERIO\_DFIFO\_DEPTH=256, OTG\_DFIFO\_DEPTH=4096

Bidir EP/Host Channel	2/4	3/6	4/8	5/10	6/12	7/14	8/16	9/16	10/16	11/16	12/16	13/16	14/16	15/16
<b>OTG-Slave-SinglePoint-NoDynamicFIFO</b>	39.91	42.61	45.89	48.84	51.44	53.99	57.78	59.75	61.5	63.18	64.67	66.49	68.25	69.84
OTG-Slv-Multipoint-DynamicFIFO	42.41	45.56	49.16	52.56	55.49	58.39	62.49	64.56	66.26	67.98	69.55	71.28	72.94	74.77
OTG-IntDMA-Multipoint-DynamicFIFO	55.07	58.82	62.54	66.73	70.21	73.72	77.4	80.33	82.24	84.28	85.96	88.21	90.04	91.88
OTG-Slv-Multipoint-DynamicFIFO-HstPeriodicsupport	42.41	45.7	49.09	52.5	55.49	58.52	62.61	64.66	66.33	67.96	69.53	71.56	73.02	74.99
OTG-IntDMA-Multipoint-DynamicFIFO-HstPeriodicSupport	54.88	58.86	62.49	66.46	70.13	73.7	77.71	80.08	82.03	84.17	85.9	88.31	90.38	91.94
OTG-DescDMA-NoDyn-Multipoint-DynamicFIFO-Host Periodic Support	75.2	79.74	85.16	90.93	95.37	100.02	105.11	107.5	110.99	113.88	116.38	119.5	121.92	125.44
OTG-DescDMA-Dyn-Multipoint-DynamicFIFO-Host Periodic Support	76.34	81.32	86.82	92.27	96.88	101.47	106.26	109.22	112.12	114.7	117.71	120.74	123.34	126.35

**Table B-10 OTG Mode with Dedicated Endpoint FIFO Area Per Configuration Per Endpoint**

Parameter	Additional Area of One Bidirectional Endpoint and Two Host Channels	Additional Area of One Bidirectional Endpoint
OTG-Slave-SinglePoint-NoDynamicFIFO	2.98K	1.72K
OTG-Slv-Multipoint-DynamicFIFO	3.35K	1.75K
OTG-IntDMA-Multipoint-DynamicFIFO	3.72K	2.07K
OTG-Slv-Multipoint-DynamicFIFO-HstPeriodicsupport	3.37K	1.77K
OTG-IntDMA-Multipoint-DynamicFIFO-HstPeriodicSupport	3.81K	2.03K
OTG-DescDMA-NoDyn-Multipoint-DynamicFIFO-HstPeriodicSupport	4.99K	2.9K
OTG-DescDMA-Dyn-Multipoint-DynamicFIFO-HstPeriodicSupport	4.99K	2.87K

**Table B-11 Device-Only Mode Area for Various Configurations: Dedicated FIFO**

Main Parameters: OTG\_MODE=4, OTG\_TX\_DINEP\_DFIFO\_DEPTH\_n=1024, OTG\_HSPHY\_INTERFACE=1, OTG\_FSPHY\_INTERFACE=0 =1, OTG\_TRANS\_COUNT\_WIDTH=11 OTG\_PACKET\_COUNT\_WIDTH=4, OTG\_RX\_DFIFO\_DEPTH=256 OTG\_TX\_NPERIO\_DFIFO\_DEPTH=256 OTG\_DFIFO\_DEPTH=4096

Bidir EP	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Device-Slave-NoDynamicFIFO	26.15	28	29.88	31.95	33.59	35.49	37.19	39.39	41.03	42.5	44.35	45.93	47.63	49.13
Device-Slave-DynamicFIFO	27.4	29.24	31.06	33.08	34.86	36.64	38.44	40.68	42.2	43.82	45.58	47.19	48.89	50.46
Device-IntDMA-DynamicFIFO	37.09	39.48	41.61	43.92	46.07	48	50.53	52.76	54.78	56.5	58.71	60.88	62.65	64.53
Device-DescDMA-DynamicFIFO-	52.67	55.93	59.1	62.14	64.93	67.68	70.48	73.31	76.33	78.25	81.35	83.67	86.62	88.6

**Table B-12 Device-Only Mode with Dedicated Endpoint FIFO Area per Configuration per Endpoint**

Parameter	Additional Area of One Bidirectional Endpoint
Device-Slave-NoDynamicFIFO	1.77K
Device-Slave-DynamicFIFO	1.77K
Device-IntDMA-DynamicFIFO	2.11K
Device-DescDMA-DynamicFIFO	2.76K

## B.2 PMU Area Numbers

[Table B-13](#) shows the DWC\_otg gate count with hibernation enhancement.

**Table B-13 DWC\_otg Gate Count with Hibernation**

Module Name	UTMI	ULPI	FS_Ded	FS_Ded/UTMI/ ULPI/HSIC/ IC_USB	DP with UTMI	FS_DedUTMI/ ULPI/HSIC/IC_USB with ADP
DWC_otg_pmu	2667.74	1928.99	1863.49	3657.25	5122.01	6307.76
DWC_otg_pmu_ahbslv	954	748.25	759.75	1066	1720.25	1825
DWC_otg_pmu_utmi	489	0.00	0.00	446	489	446
DWC_otg_pmu_fs_ded	0.00	0.00	153	280	0.00	266
DWC_otg_pmu_ulpi	0.00	311.5	0.00	312.5		312.5
DWC_otg_pmu_adp_ctl	0.00	0.00	0.00	0.00	1196	1194.75
DWC_otg_pmu_adp_ctl_if	0.00	0.00	0.00	0.00	419.25	419.75



**Note** When extended hibernation is enabled, there is approximately 5K increase in gate count.

## B.3 Estimating Area for Your Configuration

The following example shows how to estimate area for a sample configuration with 1 control and 4 bidirectional endpoints/8 host channels, 8 KB RAM (2,048 DWORDs) and periodic transfer support. Configuration parameters set during coreConsultant configuration are as follows:

OTG_MODE = 0	OTG_DFIFO_DEPTH = 2048
OTG_ARCHITECTURE = 2	OTG_DFIFO_DYNAMIC = 1
OTG_SINGLE_POINT = 0	OTG_NPERIO_TX_QUEUE_DEPTH = 8
OTG_HSPHY_INTERFACE = 2	OTG_PERIO_TX_QUEUE_DEPTH = 8
OTG_HSPHY_DWIDTH = 2	OTG_TOKEN_QUEUE_DEPTH = 0
OTG_NUM_EPS = 4	OTG_TRANS_COUNT_WIDTH = 19
OTG_NUM_PERIO_EPS = 4	OTG_PACKET_COUNT_WIDTH = 11
OTG_NUM_CRL_EPS = 0	OTG_RM_OPT_FEATURES = 0
OTG_NUM_HOST_CHAN = 8	OTG_MIN_AHB_FREQ_LESS_THAN_60 = 1
OTG_EN_PERIO_HOST = 1	*_FILTER = 0

Area for the configuration above is = 57.12K (Column 4 of Tables C-1 BiDir EPs configuration, last row)

+ 1.65K (ULPI Wrapper area from [Table B-8](#))

+ 3 \* 0.03K \* 4 (3 Additional bits of FIFO address over [Table B-1](#) configuration [from 256 to 2048]; 0.03K area per FIFO-address bit/BiDir EP from [Table B-8](#))

+ (11-4) \* 0.06K \* 4 (9 = Additional bits in Packet Counter - over [Table B-1](#) configuration, area per bit /BiDirEP from [Table B-8](#))

+ (19-12) \* \* 0.06K \* 4 (8 Additional bits in Transfer Counter - over [Table B-1](#) configuration, area per bit/BiDirEP from [Table B-8](#)) = 62.49k

[Table B-14](#) shows the additional area for DFT for 99.5% scan coverage.

**Table B-14 Area Differences Due to Scan Ready and Clock Gating**

**Main Parameters:** OTG\_MODE = 0, OTG\_ARCHITECTURE = 2, OTG\_SINGLE\_POINT = 0, OTG\_HSPHY\_INTERFACE = 1, = 2, OTG\_NUM\_EPS = 6, OTG\_NUM\_PERIO\_EPS = 4, OTG\_NUM\_CRL\_EPS = 0, OTG\_NUM\_HOST\_CHAN = 14, OTG\_EN\_PERIO\_HOST = 1, OTG\_DFIFO\_DEPTH = 2048, OTG\_DFIFO\_DYNAMIC 1, Rx/Tx FIFO = 2048, Periodic FIFO = 768, Rx/Tx QUEUE = 8, OTG\_TOKEN\_QUEUE\_DEPTH = 16, OTG\_TRANS\_COUNT\_WIDTH = 19, OTG\_PACKET\_COUNT\_WIDTH = 10, OTG\_RM\_OPT\_FEATURES = 0, OTG\_MIN\_AHB\_FREQ\_LESS\_THAN\_60 = 1, \*\_FILTER = 0

Clocks: Synthesis AHB clock = 100 MHz, PHY Clock = 60 MHz

No Clock Gating, No Scan-Ready	67.54K
Clock Gating, No Scan-Ready	62.8K
No Clock Gating, Scan-Ready	76.56K
Clock Gating, Scan-Ready	71.81K

## B.4 Speed

Table B-15 shows the areas for different frequencies and tools.

**Table B-15 Area vs. Frequency vs. Tool**

**Main Parameters:** OTG\_MODE = 0, OTG\_ARCHITECTURE = 2, OTG\_SINGLE\_POINT = 0, OTG\_HSPHY\_INTERFACE = 1, = 2, OTG\_NUM\_EPS = 6, OTG\_NUM\_PERIO\_EPS = 4, OTG\_NUM\_CRL\_EPS = 0, OTG\_NUM\_HOST\_CHAN = 12, OTG\_EN\_PERIO\_HOST 1, OTG\_DFIIFO\_DEPTH = 2048, OTG\_DFIIFO\_DYNAMIC 1, Rx/Tx FIFO = 2048, Periodic FIFO = 768, Rx/Tx QUEUE = 8, OTG\_TOKEN\_QUEUE\_DEPTH = 16, OTG\_TRANS\_COUNT\_WIDTH = 19, OTG\_PACKET\_COUNT\_WIDTH = 10, OTG\_RM\_OPT\_FEATURES = 0, OTG\_MIN\_AHB\_FREQ\_LESS THAN\_60 = 1, \*\_FILTER = 0

Frequency	Synthesis Tool	Area
AHB = 100MHz, PHY = 30 MHz	Design Compiler	67.53K
AHB = 100MHz, PHY = 60 MHz	Design Compiler	67.54K
AHB = 150 MHz, PHY = 60 MHz	Design Compiler	68.23K

## B.5 Power-Compiler Clock Gating

Table B-16 shows the percentage of flops which are clock-gatable using Power Compiler.

**Table B-16 Power Compiler Clock Gating Summary**

**Main Parameters:** OTG\_MODE = 0, OTG\_ARCHITECTURE = 2, OTG\_SINGLE\_POINT = 0, OTG\_HSPHY\_INTERFACE = 1, = 2, OTG\_NUM\_EPS = 6, OTG\_NUM\_PERIO\_EPS = 4, OTG\_NUM\_CRL\_EPS = 0, OTG\_NUM\_HOST\_CHAN = 12, OTG\_EN\_PERIO\_HOST 1, OTG\_DFIIFO\_DEPTH = 2048, OTG\_DFIIFO\_DYNAMIC 1, Rx/Tx FIFO = 2048, Periodic FIFO = 768, Rx/Tx QUEUE = 8, OTG\_TOKEN\_QUEUE\_DEPTH = 16, OTG\_TRANS\_COUNT\_WIDTH = 19, OTG\_PACKET\_COUNT\_WIDTH = 10, OTG\_RM\_OPT\_FEATURES = 0, OTG\_MIN\_AHB\_FREQ\_LESS THAN\_60 = 1, \*\_FILTER = 0

Number of clock gating elements	303
Number of gated registers	3764 (76.60%)
Number of ungated registers	1150 (23.40%)
Total number of registers	4914

The modules are coded in such a way that 75% of the flops can be clock gated using Power Compiler. The state machine and control signal flops are typically not clock gatable by Power Compiler, because they do not have enable or do not meet the minimum 3-bit common enable requirements. The power-compiler clock gating would save power even during normal operation.

In addition, the Suspend mode clock-gating/power-rail-off power optimization feature in the RTL (OTG\_EN\_PWRROPT = 1) saves additional power when the controller is in Suspend mode.

## B.6 Power Consumption

Table B-17 shows the power consumption with and without power optimization features.

**Table B-17 Power Consumption With Power Optimization Feature**

**Main Parameters:** OTG\_MODE = 0, OTG\_ARCHITECTURE = 2, OTG\_SINGLE\_POINT = 0, OTG\_HSPHY\_INTERFACE = 1, = 2, OTG\_NUM\_EPS = 6, OTG\_NUM\_PERIO\_EPS = 4, OTG\_NUM\_CRL\_EPS = 0, OTG\_NUM\_HOST\_CHAN = 12, OTG\_EN\_PERIO\_HOST 1, OTG\_DFIFO\_DEPTH = 2048, OTG\_DFIFO\_DYNAMIC 1, Rx/Tx FIFO = 2048, Periodic FIFO = 768, Rx/Tx QUEUE = 8, OTG\_TOKEN\_QUEUE\_DEPTH = 16, OTG\_TRANS\_COUNT\_WIDTH = 19, OTG\_PACKET\_COUNT\_WIDTH = 10, OTG\_RM\_OPT\_FEATURES = 0, OTG\_MIN\_AHB\_FREQ\_LESS\_THAN\_60 = 1, \*\_FILTER = 0

Simulation: AHB Frequency = 100 MHz, PHY Frequency = 60 MHz

**Note:** Power values are given in watts

	PrimetimePx No Clock Gated Netlist(A)			PrimetimePx Clock Gated Netlist(B)			A/B (Ratio)
<b>Single power rail Mode (OTG_EN_PWRROPT=0)</b>							
Module/Test	Static	Dynamic	Total (A)	Static	Dynamic	Total (B)	
All Modules/Suspend	8.32E-06	1.75E-03	1.76E-03	7.98E-06	4.06E-04	4.14E-04	4.25E+00
All Modules/Idle	8.27E-06	2.38E-03	2.38E-03	7.96E-06	6.81E-04	6.90E-04	3.45E+00
DWC_otg_wpc_slv modules/Idle	3.46E-07	1.76E-04	1.77E-04	3.46E-07	5.11E-05	5.14E-05	3.44E+00
All Modules/Normal Traffic	8.34E-06	2.39E-03	2.40E-03	8.07E-06	6.99E-04	7.08E-04	3.39E+00
DWC_otg_wpc_slv modules/Normal Traffic	3.50E-07	1.77E-04	1.78E-04	3.52E-07	5.20E-05	5.37E-05	3.31E+00
<b>Partial Power Down (OTG_EN_PWRROPT=1)</b>							
Module/Test	Static	Dynamic	Total	Static	Dynamic	Total (D)	A/D (Ratio)
All Modules/Suspend	7.96E-06	2.11E-04	2.19E-04	7.77E-06	6.21E-05	6.99E-05	3.13E+00
PHY Domain Modules/Suspend	0.000	0.05	0.05	0.000	0.05	0.05	15.0
<b>Hibernation (OTG_EN_PWRROPT=2 With UPF Support)</b>							
Module/Test	Static	Dynamic	Total	Static	Dynamic	Total (D)	A/D (Ratio)
All Modules/Suspend	2.57E-07	1.66E-05	1.69E-05	1.55E-07	1.00E-05	1.02E-05	1.66E+00

Idle and Suspend mode differ in that the PHY clock is on in Idle mode and off in Suspend mode.

The power for the PHY clock domain modules are also separately included in [Table B-17](#), so that power can be estimated at frequencies other than at 100-MHz AHB and 60-MHz PHY.



## C

## DWC\_otg LPM Add-On Feature



**Note** This is an add-on feature that requires an additional DWC-HSOTG-LPM license.

The following low-power option bits are available for programming LPM entry and exit:

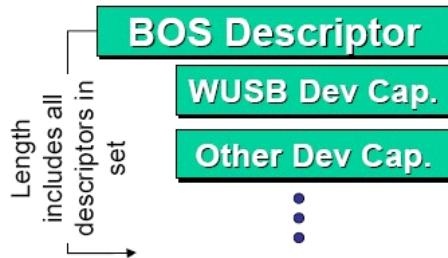
- GLPMCFG.HIRD\_Thres[4]=1 to enable Deep Sleep mode
  - Device Mode:** utmi\_l1\_suspend\_n is asserted when the received HIRD value carried in the TOKEN is greater than or equal to GLPMCFG.HIRD\_Thres[3:0]. The GLPMCFG.HIRD\_Thres[4] must be set to 1'b1.
  - Host Mode:** utmi\_l1\_suspend\_n is automatically asserted
- GLPMCFG.EnblSlpM to enable PHY Shallow Sleep mode
  - Host and Device Mode:** utmi\_sleep\_n is asserted when utmi\_l1\_suspend\_n cannot be asserted, either because the software has disabled the HIRD\_Thres[4] bit or the received HIRD is too small.
- PCGCCTL.Enbl\_L1Gating to enable controller PHY clock gating
  - Host and Device Mode:** The controller initiates PHY clock gating automatically when utmi\_l1\_suspend\_n cannot be asserted, either because software has disabled the HIRD\_Thres[4] bit or the received HIRD is too small.



- The application cannot put the controller (acting as a Host) into hibernation or partial power-down mode when it is in L1 state.
- To support Audio Class 3.0, the controller must be programmed to accept an L1 Request even when the Tx FIFOs related to ISOC, BULK, and INTR endpoints are not empty. For more details, see GLPMCFG.LPM\_Accept\_Ctrl register field description.

## C.1 Device Mode

The application software maintains the BOS descriptor defined in Section 7.4.1 of the *Wireless USB Specification*, Rev 1.0 and the USB Device LPM capability defined in Section 3 of the LPM ECN.



The device controller forwards the get\_descriptor command with BOS parameter to the application software to which, based on the database, appropriate Extension Descriptor should be formatted by the application and sent back to host in the form described in Table 3.1 of the LPM ECN.

The software determines device controller LPM capability by checking the OTG\_ENABLE\_LPM bit in User HW Config3 Register (GHWCFG3).

If OTG\_ENABLE\_LPM is read as 1'b1, the application sets LPMCap bit in the Controller LPM Configuration Register (GLPMCFG) to enable device's LPM functionality.

The controller provides a status bit, GLPMCFG.L1ResumeOK, which is set to indicate that the sleep state  $T_{L1Residency}$  has elapsed.

The application software can perform a device-initiated resume by writing to the LPM Remote Wakeup Signaling bit (DCTL.RmtWkUpSig). The controller allows device-initiated resume only if bRemoteWake was set in the previous transaction. Before this, if the controller had enabled clock gating for the controller, it must be reset at PCGCCTL.Enbl\_L1Gating. The application software must check for GLPMCFG.L1ResumeOK and confirm that it is set before setting DCTL.RmtWkUpSig. The controller then drives resume for 50  $\mu$ s and transitions into a normal mode of operation (This is similar to a normal resume, except that the DCTL.RmtWkUpSig does not need to be driven for exactly 50  $\mu$ s. The controller automatically clears DCTL.RmtWkUpSig for device-initiated resume in L1). The controller asserts the Resume/Remote Wakeup Detected Interrupt (GINTSTS.WkUpInt) at the end of the remote wakeup sequence.

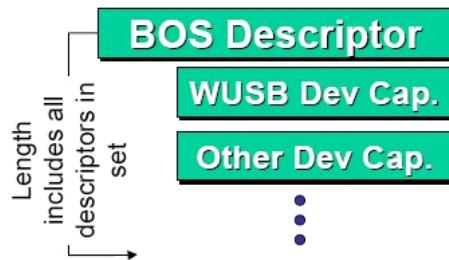
The controller can detect a host-initiated resume as long as it is in the Sleep state. The controller asserts Resume/Remote Wakeup Detected Interrupt (GINTSTS.WkUpInt) in response to a host-initiated resume, and transitions into a normal mode of operation (similar to a normal resume).

## C.2 Host Mode

The software determines host controller LPM capability by checking OTG\_ENABLE\_LPM bit in the User HW Config3 Register (GHWCFG3).

If OTG\_ENABLE\_LPM is read as 1'b1, the application sets the LPMCap bit in the Controller LPM Configuration Register (GLPMCFG) to enable host LPM functionality.

The application software queries the LPM capability of the device to which it is connected using the BOS descriptor defined in Section 7.4.1 of the *Wireless USB Specification*, Rev 1.0 and the USB Device LPM capability defined in Section 3 of LPM ECN.



The host controller forwards the result of the get\_descriptor command with BOS parameter to the application software in the form described in Table 3.1 of the LPM ECN, and the application software uses this input to check device LPM capability.

The application software must not issue an LPM transaction on a local device or an extension PID transaction on a hub connected to a remote device unless the device/hub supports LPM.

## C.3 Device-Only LPM Functions

HS OTG LPM functions are classified as either device-only or host-only. This section discusses the device-only functions in detail.



**Note** The OTG\_ENABLE\_LPM coreConsultant-configurable parameter must be 1'b1 for these functions to be active.

### C.3.1 Support for Link Power Management State

The device supports USB Link Power Management states (See Table 1.1 of *USB 2.0 Link Power Management Addendum*) by adding the new L1 (Sleep) state. There is no impact on existing OTG-related functions, such as HNP and SRP, which are a part of the Suspend state (L2), because the Sleep state cannot be entered from the Suspend state and vice-versa.

If the host signals a reset in the Sleep (L1) state, the device enters the ON (L0) state.

### C.3.2 Functions to Support L1 Entry

This section discusses the functions that support L1 Entry.

#### C.3.2.1 USB 2.0 Extension Transaction to Support LPM

The controller responds to USB 2.0 extension transactions only if the application software sets the LPMCap bit in the Controller LPM Configuration Register (GLPMCFG). This bit is common to both the host and the device. When the application sets the LPMCap bit, it can also enable low-power options by setting GLPMCFG.EnblSlpM (enabling Sleep mode), PCGCCTL.Enbl\_L1Gating (enabling PHY clock gating), or GLPMCFG.HIRD\_Thres (enabling Shallow Sleep mode).

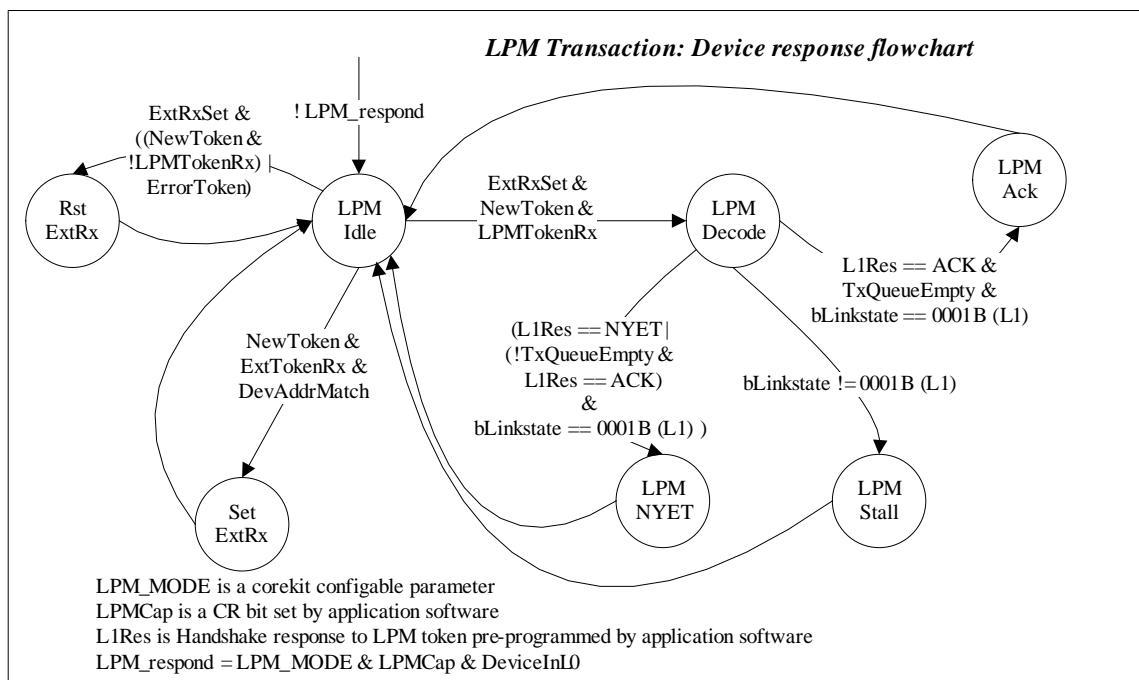
The handshake response to LPM transaction is pre-programmed by the device application software using AppL1Res in the Controller LPM Configuration Register (GLPMCFG) to ACK or NYET.

The device's response in each of these scenarios is described in the following sections.

#### C.3.2.2 Controller Handshake Response to LPM Transaction

The device response flowchart is shown in [Figure C-1](#).

**Figure C-1 LPM Transaction: Device Response Flowchart**



The LPM transaction consists of the following three packets:

1. Token packet with EXT PID 0000B from the host
2. Extended token packet with SubPID 0011B (LPM token) from the host
3. Handshake packet from the device

The LPM Transaction Response Unit is active (**LPM\_Respond = 1**) and the **ExtRxSet** flag is reset when the following conditions are false. The remainder of the LPM transaction is ignored and no response (ERROR response) is sent either on the USB or to the application.

- There is a reception error
  - PID error
  - CRC5 error
- Either the LPM-Capable bit (GLPMCFG.LPMCap) or the LPM Mode bit (GHWCFG3.OTG\_ENABLE\_LPM) is not set.
- The device is in Suspend mode (L2).
- The device is not yet reset or enumerated (L3).
- The device is already in Sleep mode (L1).

If none of the above is true, the device infers that a new token has been received and responds to the EXT PID token or the LPM token.

### C.3.2.2.1 Device Response to Token Packet With EXT PID 0000B

When the device receives a token packet, the device parses the token packet, compares the received PID with EXT PID, and compares the received ADDR field with the device ID programmed during enumeration.

- If there is an address and PID match and ExtRxSet is 0, the device sets the ExtRxSet flag and waits for the next token.
- If there is no address or PID match, the device resets the ExtRxSet flag. The device ignores the remainder of the LPM transaction and sends no response, either on the USB or to the application.

### C.3.2.2.2 Device Response to Extended Token Packet With SubPID 0011B (LPM Token)

The device responds to an LPM token only if the ExtRxSet flag is set. Setting the ExtRxSet flag indicates that the EXT PID token was successfully received, that no other packet was received afterward, and that no error conditions (as defined in “[Device Response to Token Packet With EXT PID 0000B](#)” on page 719) are present.

The device thus parses the packet following the token packet with EXT PID and takes following actions:

The device thus parses the current packet received after a token packet with EXT PID, then takes one of the following actions, depending on the type of token received.

If the EXT PID packet is repeated (ERROR)

If the next packet following the token packet with EXT PID is not an LPM token packet with LPM subPID, the device ignores the LPM transaction and sends no response.

If the next packet following the token packet with EXT PID is an EXT PID token packet, the controller considers it a new LPM transaction and sets the ExtRxSet flag.

After managing the ERROR response, the LPM Transaction Response Unit reverts to the Idle state, where it is ready to receive a new LPM transaction.

#### Token with EXT PID followed by another token packet other than LPM/EXT token

Because the received packet is not an EXT PID token, the controller resets the internal ExtRxSet flag, and the LPM transaction unit takes no other action. If the received packet is a valid non-LPM or non-EXT token packet, the controller responds to it normally; otherwise, the device infers that an LPM token was received after the EXT token. It then sets a LPMTokenRx flag and decodes the LPM token. After decoding, the response can be STALL, NYET or ACK.

## STALL Response

When there is no ERROR response, the device gives a STALL response under the following condition:

- The received LPM token's bLinkState is anything other than Sleep (L1), in other words, not 0001B.

The device sends a STALL handshake packet as a response on the USB.

The controller updates the following status registers:

- GLPMCFG.CoreL1Res is set to STALL and GLPMCFG.SlpSts is set to 1'b0.
- GLPMCFG.HIRD and GLPMCFG.bRemoteWake are updated with the value received in the LPM token.
- GINTSTS.LPM\_Int is set, irrespective of the GINTMSK.LPM\_IntMsk value, indicating that an LPM transaction was received.

### Interrupt

- If GINTMSK.LPM\_IntMsk is not set, the controller issues an application interrupt.

After catering to the STALL response, the LPM Transaction Response Unit reverts to the Idle state, where it is ready to receive a new LPM transaction.

## NYET Response

When there is no ERROR or STALL response, the device gives a NYET response under the following conditions:

- AppL1Res is programmed as NYET.
- AppL1Res is programmed as ACK, but one or more transmit queues are not empty (default).
- If the GLPMCFG.LPM\_Accept\_Ctrl[0] bit is set and one or more non-INTR transmit FIFOs are not empty.
- If the GLPMCFG.LPM\_Accept\_Ctrl[1] bit is set and the controller is in the middle of a control transfer stage.
- If the GLPMCFG.LPM\_Accept\_Ctrl[2] bit is set and one or more non-ISOC transmit FIFOs are not empty.
- If the GLPMCFG.LPM\_Accept\_Ctrl[3] bit is set and one or more non-Bulk transmit FIFOs are not empty.
- Received HIRD is greater than GLPMCFG.HIRDThres and DCTL.DeepSleepBESLReject is set.

In both the above-mentioned scenarios, a NYET handshake packet is sent as response on the USB.

Status registers are updated and an interrupt is issued, as with the STALL response in the previous section, except that GLPMCFG.CoreL1Res is set to NYET instead of STALL.

After handling the NYET response, the LPM Transaction Response Unit reverts to the Idle state, where it is ready to receive a new LPM transaction.

## ACK Response

When there is no ERROR, STALL, or NYET response, and AppL1Res is programmed as ACK, the device provides an ACK response under the following conditions:

Even though GLPMCFG.AppL1Res to ACK is pre-programmed, the controller responds with an ACK only on a *successful* LPM transaction.

The LPM transaction is successful if:

- There are no PID/CRC5 errors in either the EXT token or the LPM token, else ERROR.
- A valid bLinkState = 0001B (L1) is received in the LPM transaction, else STALL.
- If any of the GLPMCFG.LPM\_Accept\_Ctrl[3:0] bits is not set and no data is pending in the transmit or receive FIFOs, then ACK is sent, else NYET.

If the GLPMCFG.LPM\_Accept\_Ctrl[0] bit is set, then even if there is data pending in the INTR transmit FIFOs, ACK is sent, else NYET.

If the GLPMCFG.LPM\_Accept\_Ctrl[1] bit is set, then the controller is not in the middle of any incomplete control transfer stages, ACK is sent, else NYET.

If the GLPMCFG.LPM\_Accept\_Ctrl[2] bit is set then even if there is data pending in the ISOC transmit FIFOs, ACK is sent, else NYET.

If the GLPMCFG.LPM\_Accept\_Ctrl[2] bit is set then even if there is data pending in the BULK transmit FIFOs, ACK is sent, else NYET.

After handling the ACK response, the LPM Transaction Response Unit triggers the L0-to-L1 Transition Unit. The LPM Transaction Response Unit reverts to the Idle state, where it is ready to receive a new LPM transaction.

### C.3.3 Transition from L0 to L1

After sending an ACK response, the device transitions into the Sleep state. If the device's ACK response does not reach the host or hub, the host or hub may retry the LPM transaction to the device. The device must wait for this retry before transitioning to L1. The transition to the Sleep state is as follows:

1. Wait for Token Retry timeout

If an ACK is sent as response, the device waits for  $T_{L1TokenRetry} = 8 \mu s + 0.5 \mu s$  (Here, 0.5  $\mu s$  is an additional buffer provided to accommodate delay variation in the host's retry response, while 8  $\mu s$  is the LPM specification value)

- If there is any transaction from the host on the USB to this device address before the  $T_{L1TokenRetry}$  counter expires, the device resets the counter and does not transition to the L1 state.
- If the  $T_{L1TokenRetry}$  counter expires, the device continues transitioning to L1.

2. Initiate Sleep state status update

At the end of  $T_{L1TokenRetry}$  counter expiration, the device controller takes the following actions:

- Updates status registers
  - GLPMCFG.CoreL1Res is set to ACK and GLPMCFG.SlpSts is set to 1'b1

- GLPMCFG.HIRD and GLPMCFG.bRemoteWake are updated with the value received in the LPM token.
  - GINTSTS.LPM\_Int is set, irrespective of the GINTMSK.LPM\_IntMsk value, indicating that an LPM transaction was received.
  - Issues interrupt
    - If the GINTSTSMSK.LPM\_IntMsk is not set, the device controller issues an application interrupt.
  - Sends HIRD value to the device controller indicating the duration of Resume that will be transmitted on the Bus when the Host initiates LPM. The device controller provides the HIRD received from the host (through a valid LPM transaction) as an output signal for system usage.
- The following signals are used:
- The dev\_hird\_vld\_tgl signal toggles to indicate when a new hird\_value is received by the device from the host.
  - The dev\_hird\_vld\_tgl signal value is updated on dev\_hird\_rcvd signal

These signals are updated at the entry of L1 state after L1RetryTimer expires, which happens only upon successful completion of LPM transaction with ACK handshake. These signals are not updated for LPM error cases, NYET, or STALL handshakes.

### 3. Transition to SLEEP State

- The device controller changes UTMI PHY control signals to transition the bus into the L1 Sleep state. If in HS mode, the controller switches to FS mode as follows:
  - Change XcvrSelect from HS\_XVCR to FS\_XCVR.
  - Set TermSelect from HS\_TERM to FS\_TERM.
- The device controller asserts utmi\_l1\_suspend\_n (1'b0) to the UTMI PHY (Automatic PHY Deep Sleep control from the controller), if the GLPMCFG.HIRD value received from the host is greater than or equal to GLPMCFG.HIRD\_Thres[3:0] and if GLPMCFG.HIRD\_Thres[4] is set.
- If utmi\_l1\_suspend\_n cannot be asserted, the controller asserts utmi\_sleep\_n (1'b0) to the UTMI PHY (Automatic PHY Shallow Sleep control from the controller) if GLPMCFG.EnblSlpM is set.
- If utmi\_l1\_suspend\_n cannot be asserted, the device puts the controller in Power-Saving mode by disabling the PHY clock within the controller by clock gating if PCGCCTL.Enbl\_L1Gating is set.
- The device must not take more than  $T_{L1TransitionDev} = 1\mu s$  (margin of  $0.5\mu s = 10 - (8 + 1 + 0.5)$  over the LPM ECN specification value) to transition into the L1 sleep state.

#### C.3.4 L1 State Events

A (non-user) parameterizable down-counter is started ( $T_{L1Residency} = 50\mu s$ ). When this counter expires in the L1 state, the L1ResumeOK status bit is set.

The  $T_{L1Residency}$  time is essentially the PHY line states' settling time budget. The line states are valid only after this delay and the controller only looks for host-initiated resumes after it as well. Similarly, the device cannot initiate a resume until this counter has expired, because the host also must set its line states on the port after receiving an ACK response.

The device checks the line status for any line changes when GLPMCFG.L1ResumeOK is set. The device must come out of L1 whenever a reset or a resume signal is detected.

### C.3.5 Device-Initiated L1 Exit

As long as the device is in the L1 state, it can initiate L1 exit, if the device application software commands the exit by setting the LPM Remote Wakeup Signaling (DCTL.RmtWkUpSig) bit. The GLPMCFG.HIRD\_Thres[4], GLPMCFG.EnblSlp and PCGCCTL.Enbl\_L1Gating bits, if set, must be cleared before programming DCTL.RmtWkUpSig to initiate a device-initiated L1 exit.

The device application software must not set the DCTL.RmtWkUpSig bit unless the GLPMCFG.bRemoteWake status bit was set after the previously received LPM transaction token. The controller checks for GLPMCFG.bRemoteWake for device-initiated L1 exits. Thus, there can only be a transition from L1 to L0 if the GLPMCFG.bRemoteWake status bit is set.

The application software must also check that GLPMCFG.L1ResumeOK is set before setting DCTL.L1RmtWkUpSig. The hardware does not check the GLPMCFG.L1ResumeOK bit. If the software sets the DCTL.L1RmtWkUpSig when the GLPMCFG.L1ResumeOK is not set, the controller continues with L1 exit, irrespective of the value of GLPMCFG.L1ResumeOK.

The device controller immediately changes the UTMI PHY control signals to initiate an exit from the Sleep state.

The controller updates the status registers.

- GLPMCFG.CoreL1Res is set to ERROR and GLPMCFG.L1ResumeOK is set to 1'b0.
- GLPMCFG.HIRD and GLPMCFG.bRemoteWake are set back to power-on reset values.

The controller issues an interrupt.

- At the end of the device-initiated remote wakeup event, the controller clears the GLPMCFG.Slpsts bit and sets the GINTSTS.WkUpInt bit. Additionally, if the GINTMSK.WkUpIntMsk is zero, the controller issues an application interrupt.

### C.3.6 Host/Hub-Initiated L1 Exit

In the L1 state, after the  $T_{L1Residency} = 50 \mu s$  timer expires (status bit GLPMCFG.L1ResumeOK = 1), the device controller listens to the line state.

If a J-to-K transition is valid for a duration of  $J_{resume}$ :

- The device controller changes the UTMI PHY control signals to initiate an exit from the Sleep state.
- The device controller updates status registers:
  - The device controller sets GLPMCFG.CoreL1Res to ERROR and GLPMCFG.L1ResumeOK to 1'b0.
  - The device controller sets the GLPMCFG.HIRD field and the GLPMCFG.bRemoteWake bit to power-on reset values.

The controller issues an interrupt.

- At the end of the host-initiated resume event, the controller clears the GLPMCFG.Slpsts bit and sets the GINTSTS.WkUpInt bit. Additionally, if GINTMSK.WkUpIntMsk is zero, the controller issues an application interrupt.

### C.3.7 Reset Signaling-Initiated L1 Exit

The device controller listens to the line state while in L1. Reset detection is performed even during the  $T_{L1Residency} = 50 \mu s$  time period. If there is a J-to-SE0 transition valid for a time  $J_{reset}$  ( $2.5 \mu s$ ), GINTSTS.USBRst is set for reset-initiated L1 exit. The controller sets GINTSTS.USBRst, and if GINTMSK.USBRstMsk is not set, issues an application interrupt.

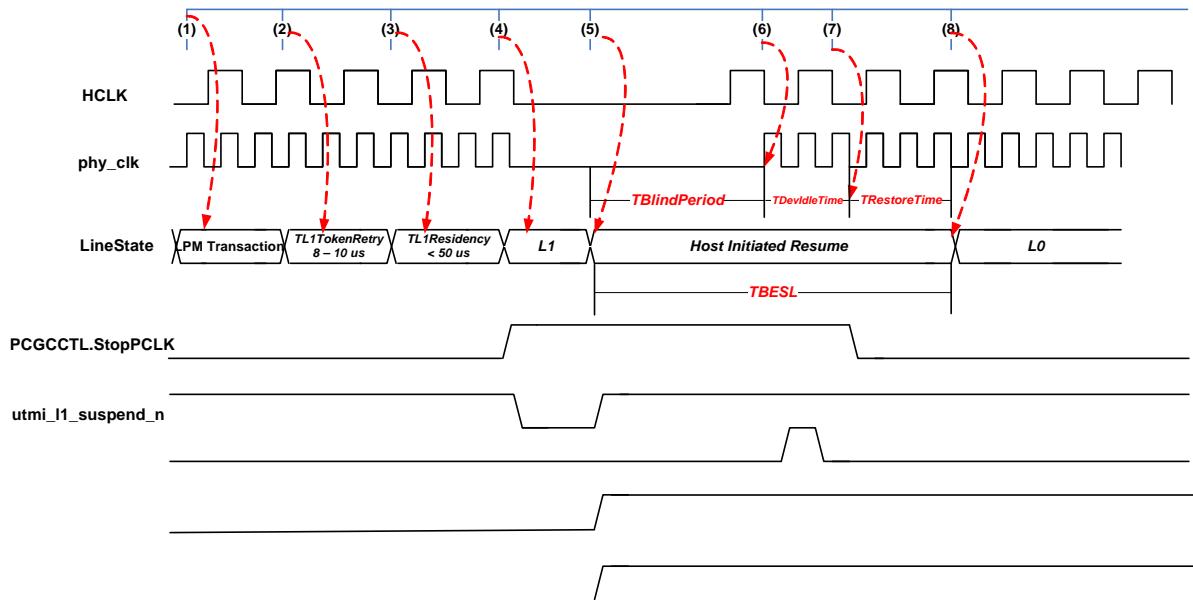


- PHY power savings: The controller puts the PHY into Deep and Shallow Sleep modes by automatically asserting utmi\_l1\_suspend\_n or utmi\_sleep\_n, respectively, while in the L1 state.
- If utmi\_l1\_suspend\_n cannot be asserted, the controller performs power-saving by blocking the PHY clock.

### C.3.8 Host/Hub-Initiated L1 Exit with EnBESL = 1'b1

[Figure C-2](#) illustrates the sequence for host/hub-initiated L1 Exit with EnBESL = 1'b1. Map the numbers listed in the figure to the description in [Table C-1](#).

**Figure C-2 Host/Hub-Initiated L1 Exit with EnBESL = 1'b1**



[Table C-1](#) explains the various stages shown in [Figure C-2](#).

**Table C-1 Stages for Host/Hub-Initiated L1 Exit with EnBESL = 1'b1**

Stage	Description
(5)	<p>The Host initiates the RESUME signaling on the LineState. The controller detects the change in LineState and does one of the following:</p> <ul style="list-style-type: none"> <li>■ If the controller is in the Deep Sleep mode, it de-asserts the utmi_l1_suspend or utmi_suspend_n signal.</li> <li>■ If the controller is in the Shallow Sleep mode, the controller de-asserts the utmi_sleep_n signal.</li> </ul>

**Table C-1 Stages for Host/Hub-Initiated L1 Exit with EnBESL = 1'b1 (Continued)**

Stage	Description
(6)	<p>When the PHY detects that the utmi_sleep_n, utmi_suspend_n, or utmi_l1_suspend_n signal is de-asserted, it starts the process of giving back the PHY clock to the controller. The PHY gives the clock to the controller in TBlindPeriod duration.</p> <p>After getting the PHY clock and HCLK clock, the controller does one of the following:</p> <ul style="list-style-type: none"> <li>■ If the controller enters hibernation, the GPWRDN.ResetDet interrupt is generated.</li> <li>■ If the controller enters the partial power-down or clock-gating state, the GINTSTS.WkUpInt interrupt is generated.</li> </ul> <p>After getting the interrupt, the application needs to initiate the restore process to the controller.</p>
(7)	The application can delay the restore process to the controller for another $T_{DevIdleTime}$ . The restore process must begin in $T_{RestoreTime}$ . Otherwise, the restore process does not complete by the time $T_{BESL}$ expires.
(8)	The device controller exits the L1 state and enters the L0 state.

### C.3.9 Reset Signaling-Initiated L1 Exit with EnBESL = 1'b1

The following table explains the various stages shown in [Figure C-2](#) for reset signaling-initiated L1 Exit with EnBESL = 1'b1.

**Table C-2 Stages for Reset Signaling-Initiated L1 Exit with EnBESL = 1'b1**

Stage	Description
(5)	<p>The Host initiates the RESET signaling on the LineState. The controller detects the change in LineState and does one of the following:</p> <ul style="list-style-type: none"> <li>■ If the controller is in the Deep Sleep mode, it de-asserts the utmi_l1_suspend or utmi_suspend_n signal.</li> <li>■ If the controller is in the Shallow Sleep mode, the controller de-asserts the utmi_sleep_n signal.</li> </ul>
(6)	<p>When the PHY detects that the utmi_sleep_n, utmi_suspend_n, or utmi_l1_suspend_n signal is de-asserted, it starts the process of giving back the PHY clock to the controller. The PHY gives the clock to the controller in TBlindPeriod duration. After getting the PHY clock and HCLK clock, the controller does one of the following:</p> <ul style="list-style-type: none"> <li>■ If the controller enters hibernation, the GPWRDN.ResetDet interrupt is generated.</li> <li>■ If the controller enters the partial power-down or clock-gating state, the GINTSTS.WkUpInt is generated.</li> </ul> <p>After getting the interrupt, the application needs to initiate the restore process to the controller.</p>
(7)	The application can delay the restore process to the controller for another $T_{DevIdleTime}$ . The restore process must begin in $T_{RestoreTime}$ . Otherwise, the restore process does not complete by the time $T_{BESL}$ expires.
(8)	The device controller exits the L1 state and enters the L0 state.



**Note** PHY power savings: The controller puts the PHY into Deep Sleep and Shallow Sleep modes by automatically asserting utmi\_l1\_suspend\_n or utmi\_sleep\_n, respectively, while in the L1 state.

## C.4 Host-Only Functions

Host-Only functions are discussed below in detail. The coreConsultant-configurable OTG\_ENABLE\_LPM parameter must be 1'b1 to enable these functions in hardware.

### C.4.1 Communication of LPM Transactions to Devices

In this databook, “remote device” refers to a device that is connected to the host port by way of a hub, as opposed to a local device, which is connected directly to the host port. The host supports the LPM feature differently for remote and local devices, as described in the following subsections.

### C.4.2 Entering the Sleep State

#### C.4.2.1 Remote Device

In a remote device configuration, the host controller commands the hub to transition a device connected on one of its ports to the Sleep state as follows:

1. Host-hub communication is accomplished through a new set-and-test port control transfer defined in the LPM ECN. [Figure C-1](#) on page 718 shows the three transaction stages in the set-and-test port feature. The hub interprets the Setup stage and sends the remote device an LPM transaction as part of the hub transaction translation function.
2. The hub returns the remote device’s response to the host in the DATA stage, through the DATA1 packet.
3. The host continues sending IN tokens in the DATA stage, to which the hub responds with either a NAK (if the LPM transaction to the remote device is incomplete) or a DATA1 (if the LPM transaction is complete).
4. A final Status stage completes the control transfer.

The host controller must support the new set-and-test port control transfer type.

The set-and-test port feature is a control-read sequence that consists of a Setup, Single-Byte (DATA1-IN), and Status (DATA1-OUT) transactions. Because all three types of transaction (SETUP, IN, and OUT) are already supported, no hardware changes are required to support set-and-test port control.

The application software prepares the data and sends all the parameters depicted in [Figure C-1](#) to the appropriate data fields.

The host responds to the set-and-test port control transfer type even if LPM support is not enabled by software.

#### C.4.2.2 Local Device

In local device configuration, the host controller commands a device to transition to the sleep state as follows:

1. The host controller issues a Control register command to initiate the LPM transaction.
2. The application software programs the following LPM transaction fields:
  - ❑ Host-Initiated Resume Duration (GLPMCFG.HIRD)
  - ❑ Host controller resume reflect duration (GLPMCFG.HIRD\_Thres) for a device-initiated remote wakeup event.
  - ❑ Local Device Remote Wake-up enable (bRemoteWake)GLPMCFG.bRemoteWake)
 and can enable the low-power options using GLPMCFG.EnblSlpM, PCGCCTL.Enbl\_L1Gating, and GLPMCFG.HIRD\_Thres[4].
3. The application software sets the Send LPM Transaction control bit (GLPMCFG.SndLPM), initiating the LPM transaction.
4. The host controller hardware generates an LPM transaction on the bus. If the response from the local device is an error, the host retries LPM transactions for the programmed number of times (GLPMCFG.RetryCnt) or until it receives a valid response from the device.
5. Completing the LPM transaction sets the GINTSTS.LPM\_Int bit. If the GINTMSK.Lpm\_Int\_Msk bit is zero, the host controller issues an application interrupt. The GLPMCFG.CoreL1Res status bit indicates the device's response for the current LPM transaction.
6. If the device response to the LPM transaction is an ACK, the host port is automatically put to sleep and the GLPMCFG.SlpSts bit is set.

## C.4.3 Entering the L1 State

### C.4.3.1 Remote Device

There have been no functional additions in hardware to implement L1 state entry in a Remote Device configuration. Because the application software queries the hub and the hub responds directly to the software, any further action (such as device status updates) can be performed in software.

### C.4.3.2 Local Device

#### Device Response Not an ACK

If the device response to the LPM transaction is STALL/NYET/ERROR, the LPM transaction status is updated in GLPMCFG.CoreL1Res. If the device response was an ERROR, the host performs LPM transactions for the programmed number of retries (GLPMCFG.RetryCnt) or until a valid response is received (ACK/STALL/NYET). The result (STALL/NYET/ERROR) is indicated to the application by setting the GINTSTS.LPM\_Int status bit. If GINTMSK.LPM\_IntMsk is zero, an application interrupt is also issued.

#### Device Response Is an ACK

For an ACK response, the controller automatically enters the L1 state as follows:

1. The host controller automatically places the local port in the Sleep state.
2. The host controller sets the GLPMCFG.SlpSts bit.
3. The host controller updates the GLPMCFG.CoreL1Res status with an ACK response.

4. The host controller indicates the response to the application by setting the GINTSTS.LPM\_Int status bit. If GINTMSK.LPM\_IntMsk is zero, it also issues an application interrupt.

The host controller performs the following actions to switch the port to the L1 Sleep state:

1. Change the UTMI control signals to the PHY to initiate the switch to the L1 state.  
If the port is in HS mode, the Host Controller switches the port to FS mode by changing the following UTMI signals:
  - ❑ Change XcvrSelect from HS\_XVCR to FS\_XCVR.
  - ❑ Change TermSelect from HS\_TERM to FS\_TERM.
2. Assert utmi\_l1\_suspend\_n (utmi\_l1\_suspend\_n = 1'b0) to the UTMI PHY (automatic PHY deep sleep control from the controller) if the GLPMCFG.HIRD\_Thres[4] is set.
3. If utmi\_l1\_suspend\_n cannot be asserted:
  - ❑ The host controller asserts utmi\_sleep\_n (utmi\_sleep\_n = 1'b0) to the UTMI PHY (automatic PHY shallow sleep control from controller) when GLPMCFG.EnblSlpM is set.
  - ❑ The Host Controller enters Power-Saving mode by gating the PHY clock internally when PCGCCTL.Enbl\_L1Gating is set.

## C.4.4 L1 State Events

### C.4.4.1 Remote Device

There are no hardware functions for L1 State events in remote device configuration.

For device-initiated L1 exit monitoring, the software must periodically poll the remote device port status using a “Get Port Status” control transfer through the hub.

- The application must interpret the additional Get Port Status meant for LPM.
  - ❑ Bit 5 of the Hub Port Status field (received through the Get Port Status Control transfer) indicates the PORT\_L1 status (that is, whether this port's link state is L1)
  - ❑ Bit 5 of the Hub Port Change field (received through the Get Port Status Control transfer) indicates a change in the associated port's link C\_PORT\_L1 status (0: No change, 1: Change)

### C.4.4.2 Local Device

The host controller waits for  $T_{L1Residencyhost}$  (50  $\mu$ s), then sets the GLPMCFG.L1ResumeOK status bit.

The  $T_{L1Residencyhost}$  time is essentially the PHY line states' settling time budget. The line states are valid only after this delay, and the host controller must look for a device-initiated resume only after this period.

Similarly, the host cannot initiate a resume until this period after entering Sleep state, because the device must also set its line states on the port after responding to the ACK response.

For device-initiated L1 exit, the host controller monitors the local port line status for any line changes after GLPMCFG.L1ResumeOK is set. If there is any change, the host controller switches the port out of sleep and generates an application interrupt.

## C.4.5 Device-Initiated L1 Exit

### C.4.5.1 Remote Device

While the device is in L1, the host application software implements a device-initiated L1 exit monitor as explained in “[Remote Device](#)” on page [728](#).

Additionally, when reading the remote device port status bits indicates a change in the PORT\_L1 status, the application software must have a means to communicate the Clear Port command. This is accomplished through a Clear Port transaction, with additional PORT\_L1 and C\_PORT\_L1 Hub Class feature selectors.

These two functions require no change to existing host controller hardware

### C.4.5.2 Local Device

While in the L1 state, the Host Controller monitors the USB line state after the  $T_{L1Residencyhost}$  period.

Whenever a J-K transition is detected, the host controller brings the local port out of L1 sleep state. The Host Controller then reflects the resume (K) back on the bus for a period specified by the GLPMCFG.HIRD\_Thres field.

The host controller updates the following status registers:

- GLPMCFG.CoreL1Res is set to ERROR GLPMCFG.L1ResumeOK is set to 1'b0
- GLPMCFG.HIRD and GLPMCFG.bRemoteWake set back to their power-on reset values

Host controller interrupt

- At the end of the Device Remote Wakeup event, the GLPMCFG.Slpsts bit is cleared and the GINTSTS.WkUpInt bit is set. If the GINTMSK.WkUpIntMsk is zero, an application interrupt is also issued.

## C.4.6 Host-Initiated L1 Exit

### C.4.6.1 Remote Device

For remote device configurations, no host controller hardware change is required, because only a ClearPortFeature must be communicated.

### C.4.6.2 Local Device

When the Port is in the L1 state, the host controller monitors the USB line state after the  $T_{L1Residencyhost}$  period for device initiated Remote Wakeup event.

Meanwhile, if the application software sets the Port L1Resume bit (GLPMCFG.PrtL1Resume), the host controller brings the local port out of the L1 sleep state.

The host controller updates the following status registers

- The host controller sets the GLPMCFG.CoreL1Res field to ERROR and the GLPMCFG.L1ResumeOK to zero.
- The host controller sets the GLPMCFG.HIRD and GLPMCFG.bRemoteWake fields back to their power-on reset values.

Host controller interrupt:

- At the end of the host resume, the GLPMCFG.Slpsts is set to zero and the GINTSTS.WkupInt bit is set. If the GINTMSK.WkUpIntMsk is zero, an application interrupt is also issued.



**Note** PHY power savings: The controller puts PHY into Deep Sleep and Shallow Sleep modes by automatically asserting utmi\_l1\_suspend\_n or utmi\_sleep\_n, respectively, while in L1.

Controller power savings: If utmi\_l1\_suspend\_n cannot be asserted, the controller performs power-saving by gating the internal PHY clock.

## C.5 Implementing LPM Over ULPI Interface

The Synopsys ULPI PHY allows writing to bit 7 of the Functional Control Register. This bit is normally reserved according to the *UTMI+ Low Pin Interface (ULPI) Specification Revision 1.1, Section 4.2.2*. A combination of the SleepM and SuspendM bits are used to drive the utmi\_sleep\_n, utmi\_l1\_suspend\_n, and utmi\_suspend\_n signals coming from the controller high or low to enable different modes of operation as explained in [Table C-3](#).

**Table C-3 Mode of Operation for Different Combinations of SleepM and SuspendM**

Bit 7 SleepM	Bit 6 SuspendM	utmi_sleep_n	utmi_l1_suspend_n	utmi_suspend_n	Mode of Operation
0	1	1	1	1	Normal operation
0	0	1	1	0	L2 Suspend
1	0	1	0	1	L1 Deep Sleep
1	1	0	1	1	L1 Shallow Sleep



If you want to use the DWC\_otg controller to support LPM over the ULPI interface, the ULPI PHY must be compatible with bit [7:6] of DWC\_otg (see [Table C-3](#)), otherwise LPM cannot be supported through the ULPI interface.

# D

## Active Clock Gating Add-On Feature

---

This appendix describes Active Clock Gating (ACG) support for the DWC\_otg controller. The following topics are discussed:

- “Active Clock Gating Feature” on page [732](#)
  - “AHB and PHY Clock Gating” on page [732](#)
  - “RAM Clock Gating” on page [735](#)
- “RAM Clock Gating Entry and Exit Scenarios” on page [735](#)
- “Programming DWC\_otg Controller for Active Clock Gating” on page [736](#)

*Related Information:*

- “Enable Active Clock Gating Support” parameter in Chapter 3, “[Configuration Parameters](#)”
- gated\_hclk\_spram and spram\_gate\_en signals in Chapter 4, “[Signal Interfaces](#)”
- PCGCCTL1 register in Chapter 6, “[Control and Status Registers](#)”



The Active Clock Gating Feature is available only in versions 4.00a and later of the controller.  
To enable this feature, you need the DWC-OTG-V4 license.

---

## D.1 Active Clock Gating Feature

Using the ACG feature allows the controller to save dynamic power by internally gating the following clocks to a subset of the modules in the controller:

- AHB clock
- PHY clock
- RAM clock

The advantages of using the ACG feature is as follows:

- Gating the AHB and PHY clock to the internal modules reduces dynamic power in the DWC\_otg (host/device) controller in no-traffic scenarios such as idle (L0 state), suspend (L1/L2 states), and resume/remote wakeup. For example, during IDLE periods in L0 when there is no USB and AHB traffic, the controller can reduce its power consumption by 50 to 70% (based on measurements from gate-level simulations for a maximum configuration).
- Gating the RAM clock output of the controller reduces the power consumption in SPRAM. The SPRAM clock is gated when all the FIFOs (RXFIFO, TXFIFO) are empty, and there is no operation from Endpoint Info Control to the SPRAM. This lowers the power consumed by the SPRAM when it is not accessed.
- When this feature is used with clock gating cells automatically inserted by the synthesis tools, the combined effect of tool-inserted clock gating and active clock gating further reduces the power consumed by the controller.

### D.1.1 AHB and PHY Clock Gating

The DWC\_otg modules are gated based on the AHB and USB bus traffic. In some USB applications, complete USB bandwidth is not utilized. In some scenarios, the controller stays in L0 idle state and does not enter L1/L2 because the duration is small. During L1/L2 entry/exit, and resume/remote wakeup, not all the modules in the design need a running clock until the traffic starts. In these cases, the AHB and PHY clock are gated internal to the controller thereby reducing the amount of dynamic power consumed by the controller during the IDLE periods.

#### D.1.1.1 AHB and PHY Clock Gating Events

Figure D-1 on page 733 shows the ACG entry and exit sequence. It illustrates the sequence of events that occur inside the DWC\_otg controller. The markers (1), (2), (3), and (4) represent the following events:

##### Entry Sequence Events:

1. The PHY clock domain is ready to be gated, but the AHB domain is not. No clock is gated.
  2. The AHB domain is ready to be gated, but the PHY clock domain is not. No clock is gated.
- Note that event (2) might not always occur after event (1). It depends on the actual sequence of events occurring on the AHB and USB buses.
3. Both the AHB and PHY clock domains are ready to be gated. Both the clocks (hclk and phy\_clk) fed to the internal modules are gated.

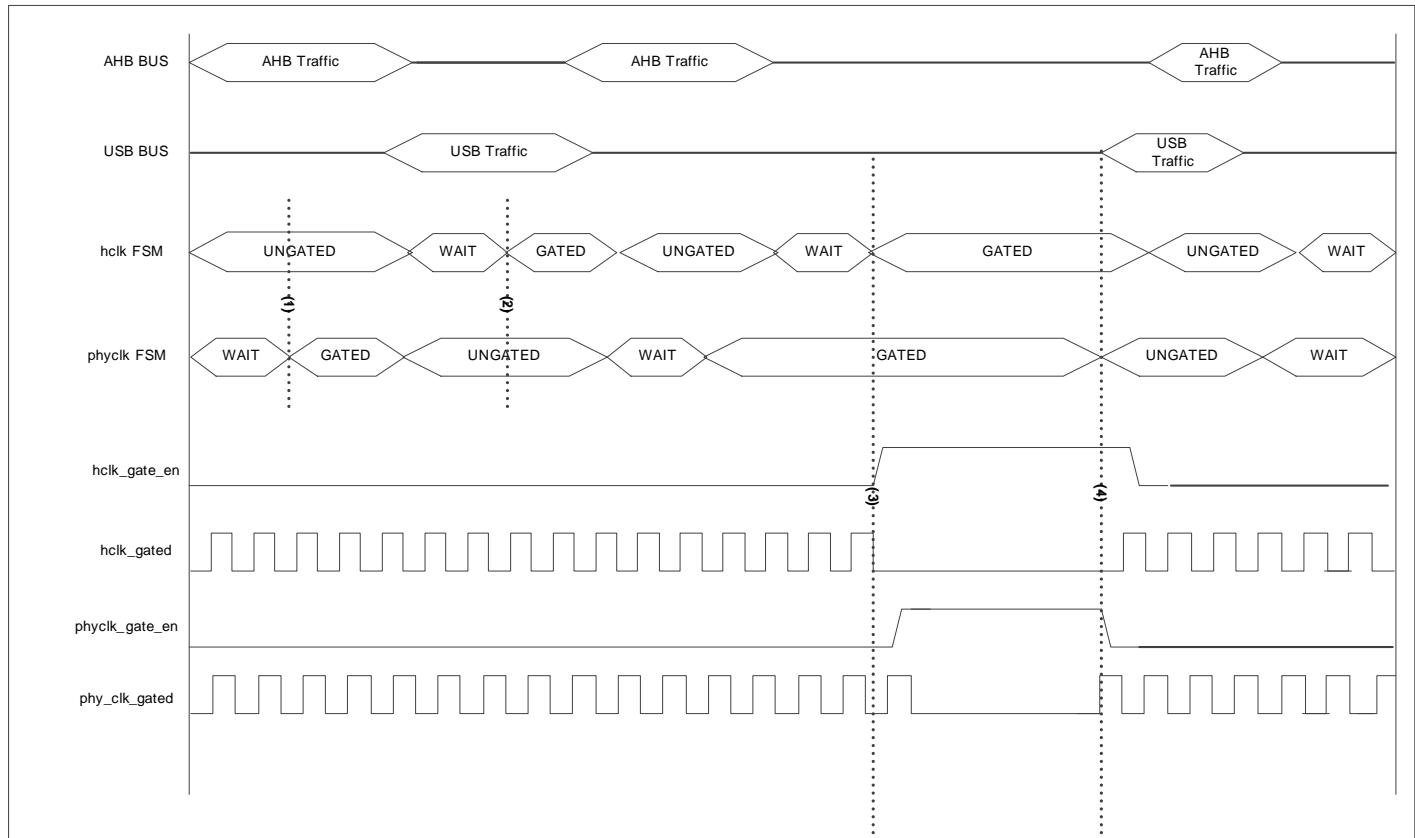
Only when both the domains do not have traffic, gating of the clocks occurs.

### Exit Sequence Event:

4. Wakeup from either the PHY clock or the AHB domain stops gating of the respective clock and synchronizes the clock in the other domain.

Note that the clock and the clock events in [Figure D-1](#) on page 733 are not to any scale, and are for illustration purposes only.

**Figure D-1 ACG Entry and Exit Sequence Events**

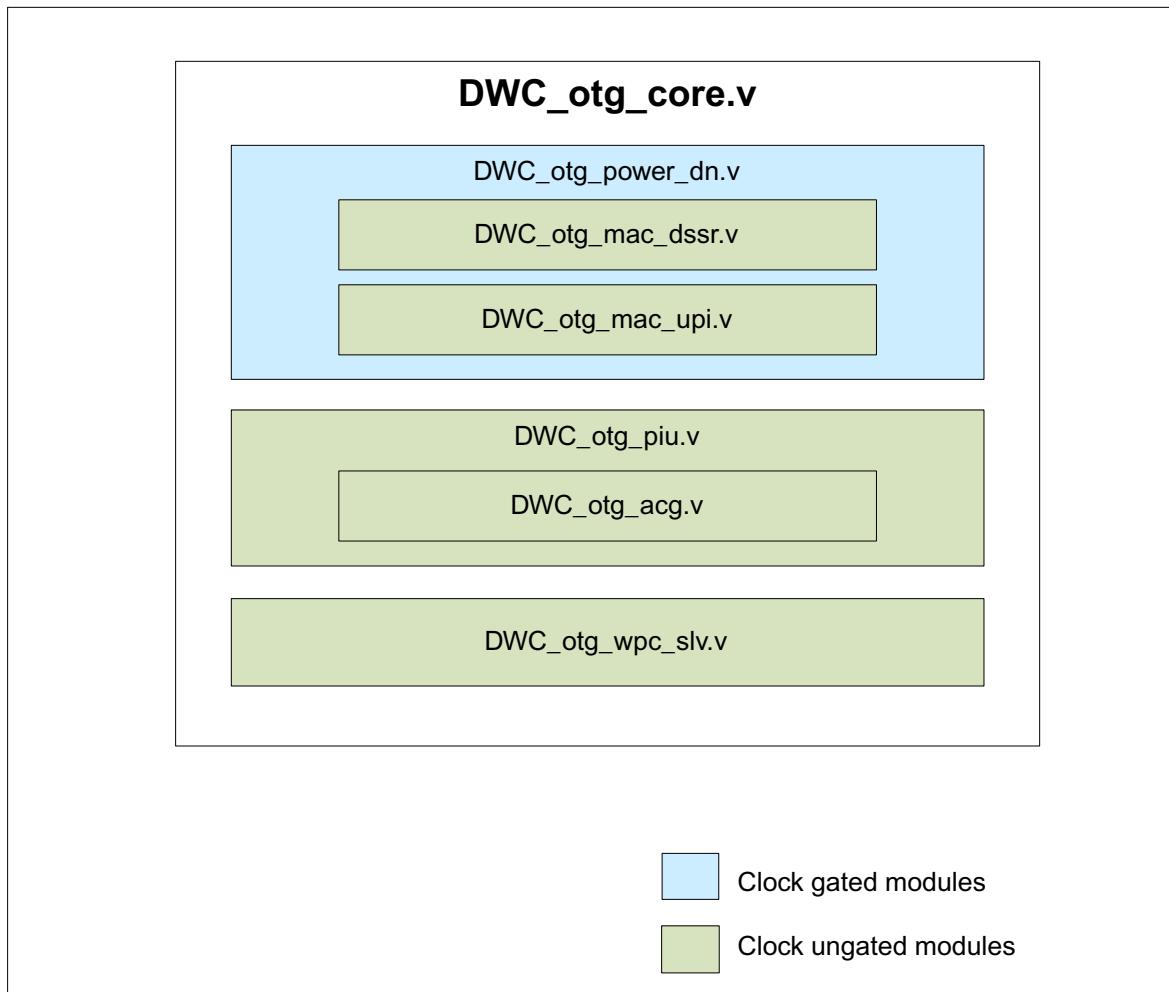


### D.1.1.2 Gated and Ungated Modules

The DWC\_otg\_acg.v module controls the AHB and PHY clock gating. This module is instantiated inside the DWC\_otg\_piuv module.

**Figure D-2** shows the modules in which the AHB and PHY clocks are gated and ungated. The modules such as DWC\_otg\_piuv, DWC\_otg\_mac\_dssrv, DWC\_otg\_mac\_upiv, and DWC\_otg\_wpc\_slv are ungated because these modules contain the wakeup logic.

**Figure D-2 Block-Level Overview Of Gated and Ungated Modules**



### D.1.1.3 Limitations of Active Clock Gating Feature

- This feature is not applicable for HNP- and SRP-Capable OTG (Device and Host) configurations (OTG\_MODE=0).

This feature is not applicable for External Hibernation configurations.

- This feature gates the AHB and PHY clock internally only during IDLE periods in between the traffic. Therefore, this feature is ideally suited for cases where traffic has a burst pattern and a large segment of the frame/microframe is IDLE.

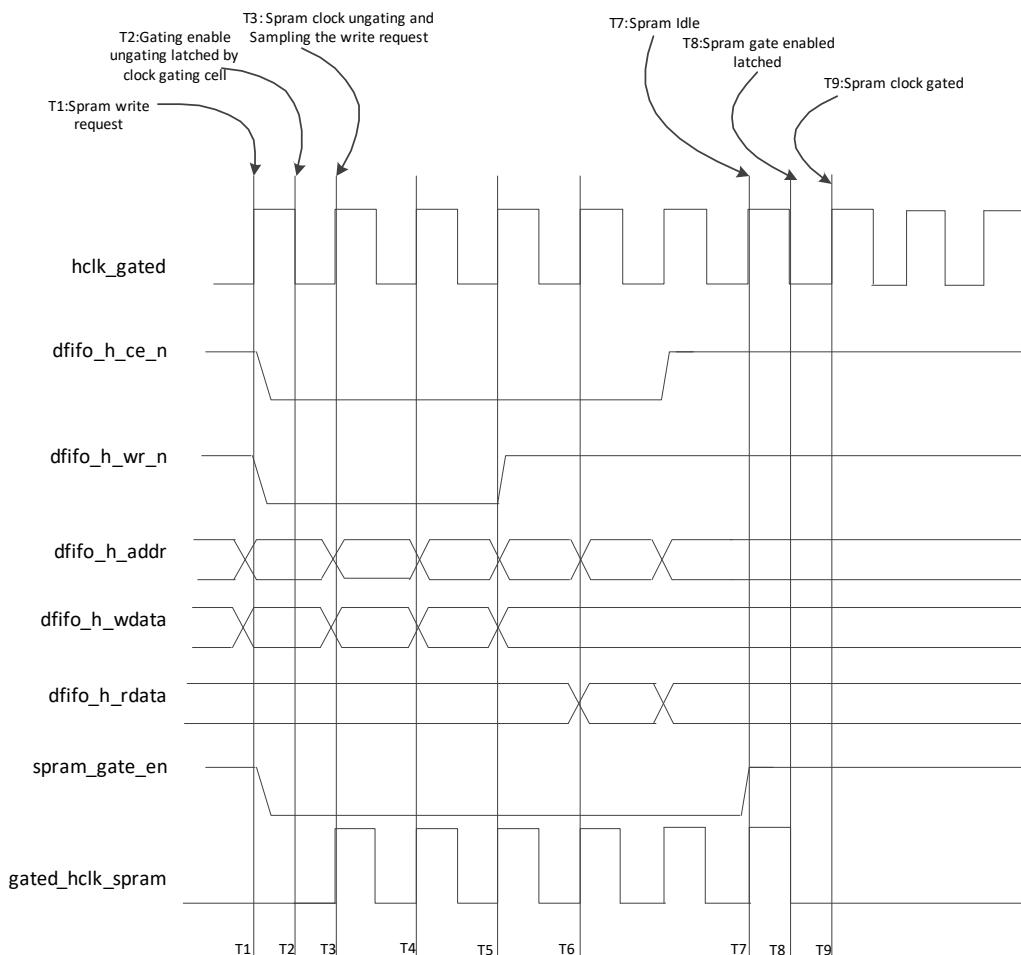
In use-cases where there is a lot of continuous traffic within a frame/microframe with little IDLE time in between traffic, the controller will not have the opportunity to gate the clocks internally and reduce its power consumption.

## D.1.2 RAM Clock Gating

In this gating method, the RAM clock is gated when the RAM/SRAM is not being accessed. With RAM clock gating, the clock for the SPRAM can be gated. The gating is done when the RxFIFO and TxFIFO are empty, and the Endpoint Info Control is not trying to access the SPRAM.

[Figure D-3](#) represents the RAM Clock Gating Entry and Exit Scenarios.

**Figure D-3 RAM Clock Gating Entry and Exit Scenarios**



In [Figure D-3](#) before timestamp T1, the RAM clock gating takes place as follows:

- T1: A RAM write request is generated, and the spram\_gate is deasserted.

- T2: The spram\_gate\_en signal is sampled by the negative level D latch, and the clock is ungated at T3.
- T3: The ungated clock is available for RAM. The RAM samples a write control signal and data in the same posedge.
- T7: There is no access to RAM.
- T8: The RAM gate enable is asserted, which is sampled by the negative level D latch.
- T9: The clock is gated.

### Ram Clock Gating Requirements

1. RAM should provide or sample the data in the same clock cycle as the read and write control signals.
2. The delay from any register for the Negative D Level Latch Clock Gating cell should be either less or equal to half the RAM clock cycle. This delay should be the same for the external clock gating cell scenario.
3. When the external clock gating cell is used, it is advantageous to use the negative level D latch clock gating cell as a clock gating cell.

## D.2 Configuring DWC\_otg Controller for Active Clock Gating

You can enable support for the ACG feature in the DWC\_otg controller by selecting “Yes” for the configuration parameter “Enable Active Clock Gating Support” in coreConsultant.



If this parameter is set, AHB and PHY clock gating needs additional programming. See [“Programming DWC\\_otg Controller for Active Clock Gating”](#).

## D.3 Programming DWC\_otg Controller for Active Clock Gating

To enable the ACG feature for gating RAM, AHB, and PHY clocks,

1. Ensure the GHWCFG4[12] register bit is set to 1.
2. Program the PCGCCTL1 register.
  - ❑ In device mode, during device initialization, after step 2, program the PCGCCTL1 register (see the “Device Initialization” section in the Programming Guide).
  - ❑ In host mode, during host initialization, after step 2, program the PCGCCTL1 register (see the “Host Initialization” section in the Programming Guide).

This step is needed to gate the RAM, AHB, and PHY clocks:

- To gate the AHB and PHY clocks, set the PCGCCTL1 register bit 0 (PCGCCTL1[0]=1'b1) to 1.
- To gate the RAM clock, set the PCGCCTL1 register bit 3 (PCGCCTL1[3]=1'b1) to 1.

# E

## Service Interval-Based Scheduling for ISOC IN Endpoints

---

This appendix describes the controller support for scheduling transfers for ISOC IN endpoints in Service Interval mode (DCTL.ServInt=1'b1). The appendix also describes the following features:

- Service Interval Mode:
  - ADC 3.0 support using ref\_clk
  - Auto Remote Wakeup Feature
- “Configuring DWC\_otg Controller for Using Service Interval-Based Scheduling for Hardware” on page [740](#).
- “Programming Configurations Available with Service Interval Mode Enabled Configuration” on page [740](#).

*Related Information:*

- Enable Service Interval-based scheduling parameter in [Chapter 3, “Configuration Parameters”](#).
- ref\_clk signal description and requirements in “” on page [170](#).
- DCTL and GREFCLK registers in “[Additional Details on GLPMCFG.HIRD\\_Thres Register Field](#)” on page [295](#).

## E.1 Service Interval Mode Feature

When this feature is enabled (DCTL.ServInt=1'b1 and DCTL.IgnrFrmNum=1'b0), the device controller holds the ISOC IN data for the given Isochronous Service Interval for this ISOC IN endpoint, that is, the data is not flushed until the last frame or microframe of the service interval. If the descriptor is fetched after the frame or the microframe number in the descriptor is elapsed, then it is closed with the BUF\_FLUSH status.

The device controller can respond with the ISOC IN Data corresponding to this service interval, if the token is received in any frame or microframe of the service interval.

The device controller can fetch the data in any number of frames/microframes before the frame/microframe programmed in the descriptor. It can fetch only two service intervals data in advance, but application can maintain descriptor list with data for more than two service intervals. However, application should follow a separate flow for scheduling transfers for binterval greater than 12. For more details, see “Scheduling Transfers for Higher bintervals (>12)” section in the Programming Guide.

The device controller does not contain any information on the binterval used for a certain endpoint. The application should keep the track of the service interval boundaries, and program the correct number of the frames or microframes into the descriptor.

For more details on scheduling ISOC IN transfers using Service interval mode, see the Programming Guide.



If the host polling rate is higher than the binterval of the Endpoint:

- When binterval < 12: The device controller can send the data of the next service interval if the data is fetched.
- When binterval > 12: The device controller can send the data of the next service interval, or resend the data of the current service interval.

It is recommended to choose the FIFO size in such a way that at least one additional packet is accommodated:

- Low bandwidth (1 packet per uF): FIFO size  $\geq$  2 MPS
- High bandwidth (3 packets per uF): FIFO size  $\geq$  4 MPS

### E.1.1 Using ref\_clk for ADC 3.0 Support

When the Service Interval feature is enabled (OTG\_SERV\_INT\_ENH=1), the controller uses a free-running external reference clock (ref\_clk) provided by the SoC to keep track of the frame and microframe counters. The Device controller can fetch and flush the data during L1.

Application should not exercise any Power Gating features to take advantage of the Service Interval mode. If Power Gating is exercised, the controller loses track of the frame and microframe counters, and the application must reschedule transfers after the frame number crosses the Host SOF.

The ref\_clk requirements are as follows:

- The period of ref\_clk must be an integer multiple of 125 us.
- The minimum ref\_clk frequency supported is 12 MHz.
- The recommended ref\_clk frequencies are 16, 17, 19.2, 20, 24, 30, and 40 MHz.

For configuring the ref\_clk, see [Chapter 6, “Control and Status Registers”](#).



**Note** The GREFCLK.RefClkMode fallback bit allows the user to use different operational modes when the Service Interval configuration parameter is enabled (OTG\_SERV\_INT\_ENH=1).

### E.1.2 Auto Remote Wakeup Feature

When the Service Interval mode is enabled and GREFCLK.RefClkMode is set, the internal frame/microframe counters are activated with ref\_clk even during L1. This allows the controller to maintain the frame/microframe numbers without Host sending the SOFs.

However, small errors due to clock uncertainty could accumulate to the clock frequency, and the frame/microframe counters can become asynchronous with the Host SOF timings, if the controller is in L1 for a sufficiently long time.

In order to prevent the controller from losing sync with the Host SOF timings, an interrupt is generated (GINTSTS2.RmtWkupAlert) to inform the application that the controller has been in L1 for a long time.

This interrupt is generated after the GREFCLK.SOF\_CNT\_WKUP\_ALERT number of frames/microframes is elapsed or counted by the controller in L1.

Upon receiving this interrupt, the application must initiate a Remote Wakeup in the device controller. The interrupt is generated sufficiently long before the ref\_clk error is added to one frame/microframe. This includes the time required to complete the Host-initiated resume.

## E.2 Configuring DWC\_otg Controller for Using Service Interval-Based Scheduling for Hardware

You can enable the support for Service Interval Feature in the DWC\_otg Controller by selecting “Yes” for the “Enable Service Interval Based- Scheduling for ISOC IN Endpoints” configuration parameter in coreConsultant.

This feature can be enabled only for the following configuration settings:

- OTG\_MODE == 2 OR OTG\_MODE == 4  
Only Non-OTG Device or Non-OTG DRD configurations can support this feature.
- OTG\_EN\_DESC\_DMA == 1  
Descriptor DMA should be enabled.
- DWC\_otg v4 license required.

## E.3 Programming Configurations Available with Service Interval Mode Enabled Configuration

[Table E-1](#) and [Table E-2](#) show the programming combinations supported when Service Interval feature is enabled (OTG\_SERV\_INT\_ENH=1).

**Table E-1 Service Interval Mode of Operation (Recommended)**

OTG_SERV_INT_ENH	Mode of Operation	GAHGCFG. DescDMAEn	DCTL. IgnrFrmNum	DCTL. ServInt	GREFCLK. RefClkMode
1	Desc/service interval	1	0	1	1

**Table E-2 Other Modes of Operation**

OTG_SERV_INT_ENH	Mode of Operation	GAHGCFG. DescDMAEn	DCTL. IgnrFrmNum	DCTL. ServInt	GREFCLK. RefClkMode
1	Desc/IgnrFrm	1	1	0	0
1	Desc/Non-IgnrFrm	1	0	0	0
1	BufDMA/PTI	0	1	0	0
1	BufDMA/Non-PTI	0	0	0	0

# F

## DWC\_otg HSIC Add-On Feature

---



**Note** This is an add-on feature that requires an additional DWC-HSOTG-HSIC license.

---

The DWC\_otg HSIC functions are classified as Device only functions and Host only functions. The common functions are supported in both device only and host only cores.

### F.1 Common Functions in Device-Only and Host-Only Cores

HSIC support can be controlled (enabled/disabled) by a coreConsultant-configurable parameter.

You can use the if\_select\_hsic input signal to switch to the HSIC mode of operation at power-up. If you use the coreConsultant configurable parameter to enable HSIC interface capability, you can use the if\_select\_hsic input signal to disable or enable this signal.

HSIC interface capability can also be disabled or enabled by software, if you enabled the parameter during coreConsultant configuration and the if\_select\_hsic signal is 1. The same controller can be configured for use with a non-HSIC interface. In this case, the USB 2.0 UTMI+/ULPI/FS interface can be optionally supported. Both the HSIC and USB 2.0 can share the UTMI.

All existing features and LPM will be part of an HSIC-compatible controller.

There is no external interface change, either on the application side or on the PHY side, though a separate PHY that supports HSIC is required.

Only the UTMI interface is supported currently for HSIC.

The USB 2.0 UTMI and HSIC UTMI interfaces are pin-shared.



**Note** The HSIC interface between the controller and the HSIC PHY is built over UTMI interface by modified interpretation of the UTMI signals. Therefore, Synopsys recommends to use Synopsys HSIC PHY with Synopsys HSIC controller. If you are using a third-party HSIC PHY with Synopsys HSIC controller, you must match the interface requirements described in “[UTMI Signaling Details](#)” on page 743.

---

## F.2 Detailed Device-Only Functions

The controller connects to a host only if software enables the GLPMCFG.HSICCon bit.

In HSIC mode, the controller enumerates only as a high-speed device.

The controller does not perform high-speed device chirps nor does it expect a host chirp during enumeration.

The controller supports device-side changes in UTMI signaling for the following scenarios. Device attach and enumeration ([“HSIC Attach and Enumeration” on page 743](#))

- Host-initiated resume ([Figure F-2 on page 745](#))
- Device-initiated remote wakeup ([“Device-Initiated Remote Wakeup” on page 746](#))

The signals that need changes are termsel, xcvsrel, opmode, txvalid, and datain.

## F.3 Detailed Host-Only Functions

In Host mode, the controller enters an enabled state only if the GLPMCFG.HSICCon bit is set. The controller does not expect a high-speed device chirp, nor does it perform a host chirp during enumeration.

The controller supports host-side changes in UTMI signaling for the following scenarios.

- Device attach and enumeration ([Figure F-1](#))
- Host-initiated resume ([Figure F-2 on page 745](#))
- Device-initiated remote wakeup ([Figure F-3 on page 746](#))

The signals that need changes are termsel, xcvsrel, opmode, txvalid, and datain (Note that pull-ups are implemented in the PHY and not in the controller)

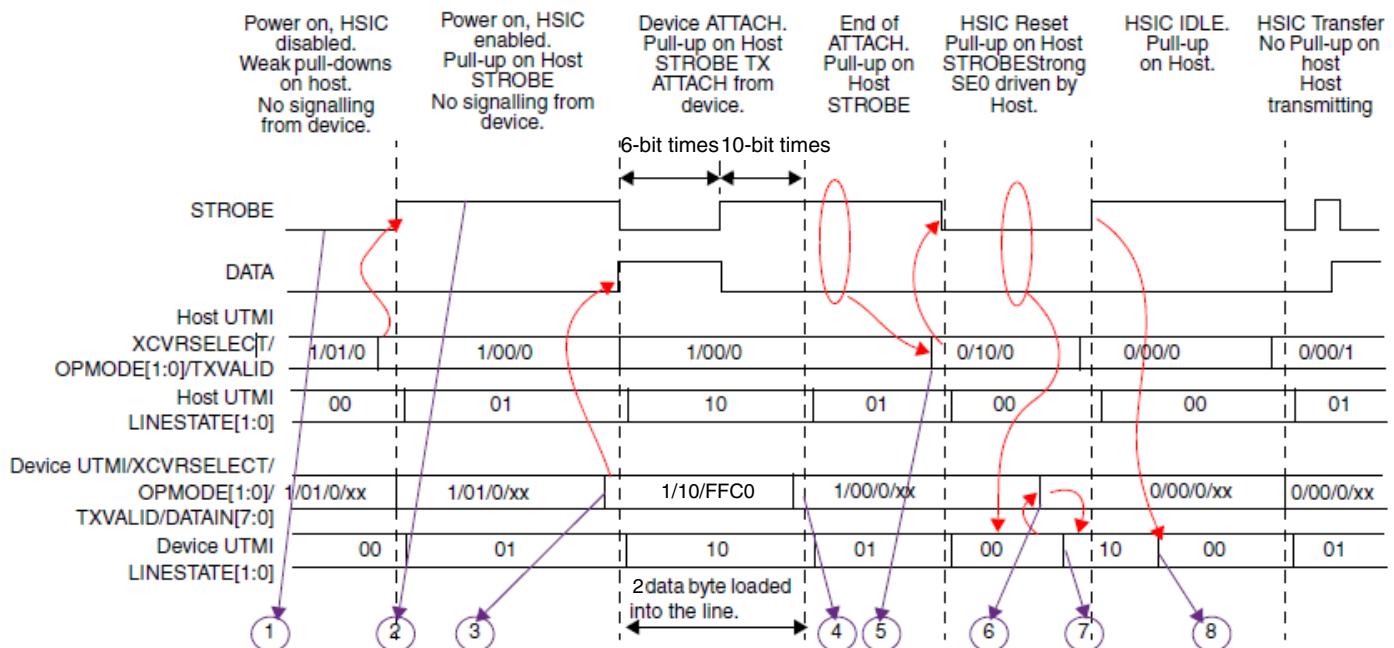
## F.4 UTMI Signaling Details

The UTMI signaling in this section applies when if\_select\_hsic is set to 1 and GLPMCFG.InvSelHsic = 0 within the controller. This enables the controller either in host mode or device mode to choose HSIC mode of operation. In this section, clock is used for hsic\_strobe signal and data is used for hsic\_data signal on the HSIC serial interface.

### F.4.1 HSIC Attach and Enumeration

[Figure F-1](#) depicts how the host- and device-side UTMI interfaces interact and its reflections on the clock and data of the PHY for attach and enumeration sequence.

**Figure F-1 HSIC Attach and Enumeration**



1. On power up, STROBE and DATA are in Power ON state but HSIC PHY is disabled.
2. When HSIC PHY is enabled, the host indicates an idle state by driving OPMODE from 2'b01 to 2'b00. The host and device detect 2'b01 on their LINESTATE.
3. The device now attaches to the system by asserting TXVALID to 1'b1 and DATAIN to 8'hC0 for one PHYCLK, and TXVALID to 1'b1 and DATAIN to 8'hFF for the next one PHYCLK. This translates into Peripheral-CONNECT in which the DATA line is high and STROBE is low.



For 30 MHz PHYCLK, the same physical handshake from a device will still apply, as of 60 MHz PHYCLK. But, the HSIC PHY will extend the idle state duration to the rest of the PHYCLK duration. The host and device in this duration detects 2'b10 on their LINESTATE.

4. The device indicates the end of attach by deasserting the TXVALID. Both the host and the device switch their LINESTATE to 2'b01. The device also switches OPMODE to 2'b00 operation. This idle period can be for one or more STROBE periods.
5. The host indicates the beginning of HSIC PHY enumeration by asserting XCVRSELECT to 1'b0 and switching OPMODE to 2'b10. This causes HSIC PHY interface switch to reset state (STROBE line low and DATA line low). The host LINESTATE detects 2'b00. The device LINESTATE also detects 2'b00.
6. On detection of 2'b00 on LINESTATE for 2.5us, the device controller indicates the end of reset by deasserting XCVRSELECT to 1'b0.
7. Once device switches to high speed (XCVRSELECT is 1'b0), any SE0 on the bus (STROBE line low and DATA line low) is indicated as 2'b10 on the device LINESTATE.
8. When the reset is complete, the host switches its OPMODE to 2'b00 to indicate the end of the HSIC PHY reset. When the bus is in HSIC PHY idle, the device LINESTATE indicates 2'b00.

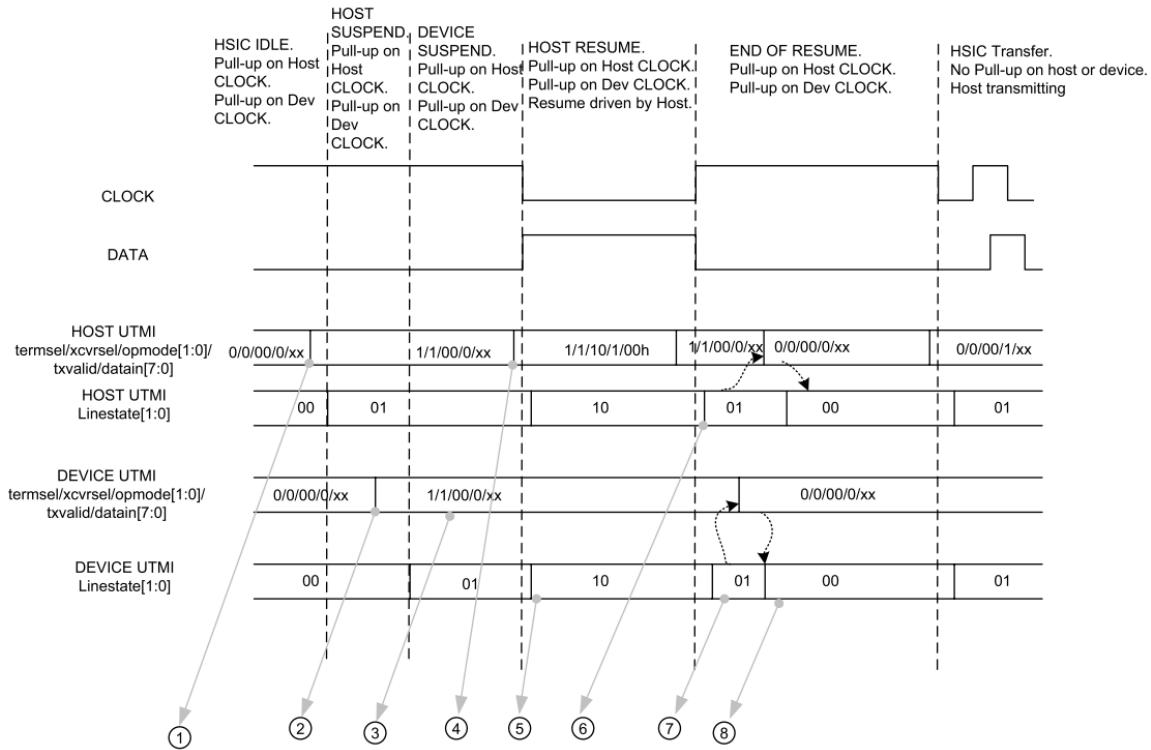
#### F.4.2 Compile Time Configuration Options

See “[HSIC Attach and Enumeration](#)” on page [743](#) for HSIC configuration.

#### F.4.3 Host-Initiated Resume

Figure F-2 depicts how the host- and device-side UTMI interfaces interact and their reflections on the CLOCK and DATA of the PHY for a host-initiated resume sequence.

**Figure F-2 Host-Initiated Resume**

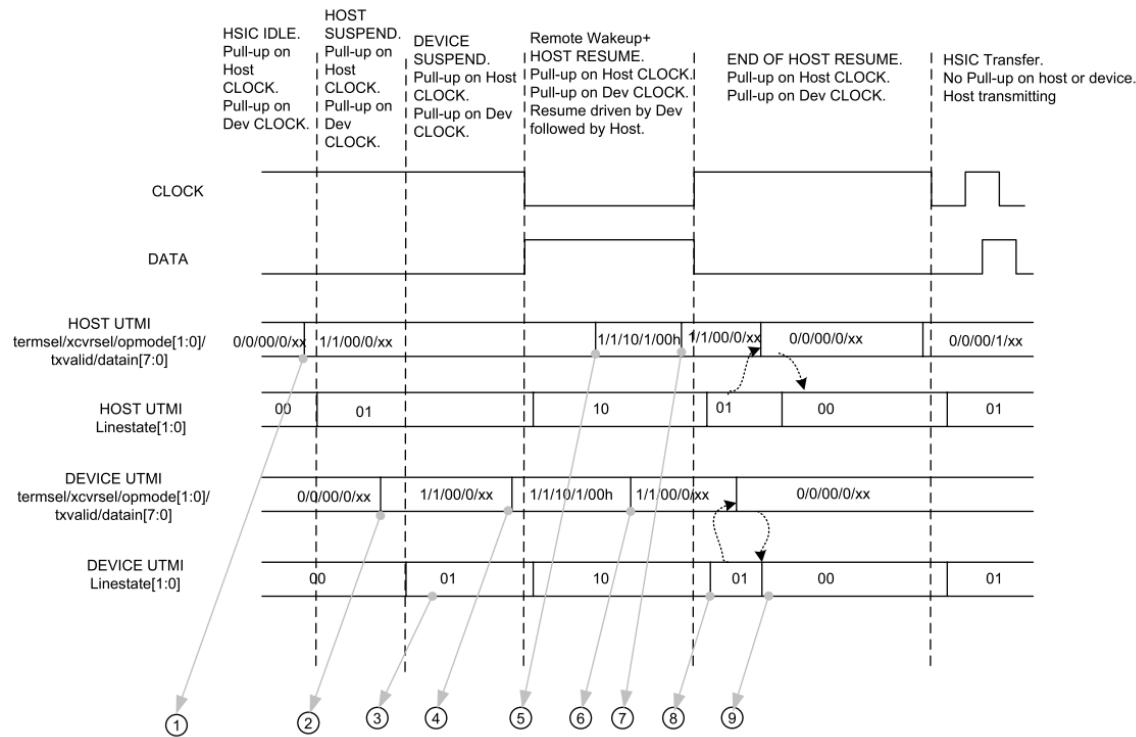


1. The host suspends the bus.
2. The device detects an early suspend (3 ms of SE0 on line state), asserts termsel = 1, xcvselsel = 1.
3. The device checks bus again after a pull up delay and detects J (ensuring that it is a suspend and not a reset).
4. The host comes out of suspend by driving a resume.
5. The device detects a resume on the bus (K on line state), and waits for the resume to end.
6. The host ends the resume.
7. The device detects end of resume (K-to-J transition on line state). This is the main difference from non-HSIC, in which the end of the resume is detected by SE0. On detecting the end of resume, the device returns to IDLE by making termsel = 0 and xcvselsel = 0.
8. Eventually, the line state returns to SE0.

#### F.4.4 Device-Initiated Remote Wakeup

Figure F-3 depicts how the Host and Device side UTMI interfaces will interact and its reflections on the CLOCK and DATA of the PHY for a Device initiated remote wakeup sequence.

**Figure F-3 Device-Initiated Remote Wakeup**



1. Host suspends bus.
2. Device detects early suspend (3ms of SE0 on Line state), asserts termsel=1, xcvsels=1.
3. Device checks bus again after pull up delay and detects J (ensuring that it is a suspend and not a reset).
4. Device application sets Remote Wakeup Control bit, and device starts driving resume on bus.
5. Host detects remote wakeup and starts driving resume on the bus.
6. Device ends its resume signaling after 3ms.
7. Host ends resume.
8. Device detects end of resume (K->J transition on line state). This is the main difference from non-HSIC. In non-HSIC, the end of resume is detected by SE0. On detecting end of resume, device goes back to IDLE by making termctrl = 0, xcvsels=0.
9. Eventually line state goes back to SE0.

**G**

# DWC\_otg IC\_USB Add-On Feature



**Note** This is an add-on feature that requires an additional DWC-HSOTG-FS-ICUSB license.

DWC\_otg IC\_USB functions are classified as device-only functions and host-only functions. The device and host functions include all the device-only functions and host-only functions.

## G.1 Detailed Device-Only Functions

- Support for both self- or bus-powered peripheral
- Removable peripheral or fixed peripheral option
- Multiple voltage class
- Control of SW3, SW4, SW5 and SW6 in LS mode of operation
- Control of SW3, SW4, SW5 and SW6 in FS mode of operation
- Conversion of FS\_LS signals to IC\_USB signals

### G.1.1 Functions to Support Self- and Bus-Powered Peripherals

To support self- and bus-powered functions, the HS OTG controller uses the ic\_usb\_vbusvalid input signal. For self-powered peripherals, this input must be tied high. For bus-powered peripherals, this input must be derived from the IC\_VDD line. Circuitry external to the controller must monitor the IC\_VDD line and when the voltage is above  $V_{OP}$  ( $V_{OP} = 0.8 * IC\_VDD$  minimum) ic\_usb\_vbusvalid must be asserted. This means that  $V_{MIN}$  must also be above the required value for correct device operation. The HS OTG controller does not start connection sequence unless ic\_usb\_vbusvalid is asserted.

### G.1.2 Functions to Support Removable and Fixed Peripherals

There is no function within the HS OTG controller to support removable peripheral. System design should ensure that for a removable peripheral, ic\_usb\_vbusvalid should be low unless the device is attached to a host. By doing so, the user will still be able to support removable peripheral using the HS OTG controller.

### G.1.3 Multiple Voltage Class

To support multiple voltage class, external circuitry would be required to detect the applied voltage level. No functions are performed within the controller.

### G.1.4 Control of SW3, SW4, SW5 and SW6 in LS Mode of Operation

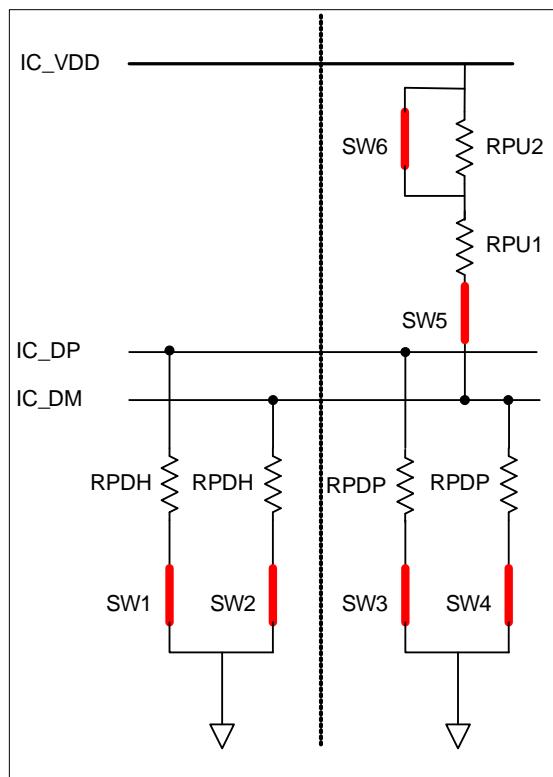
[Figure G-1](#) shows the pull-up/pull-down resistor locations for LS mode of operation.

SW1 and SW2 are controlled by the host, whereas SW3, SW4, SW5, and SW6 are controlled by the device.

The HS OTG controller has individual controls for the four device-controlled resistors.

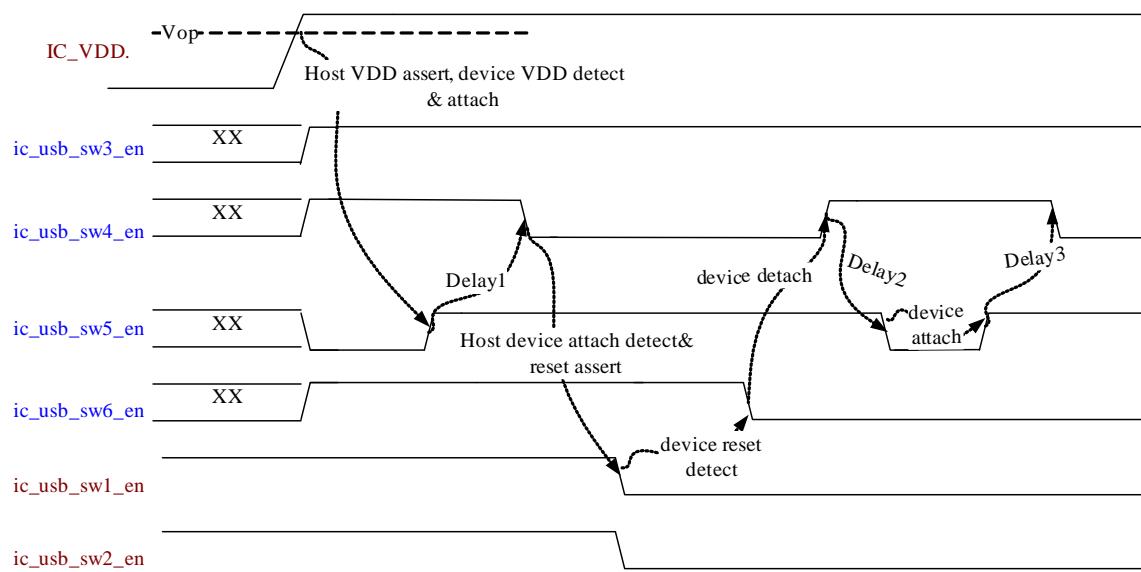
When `ic_usb_sw3_en` is deasserted (low), SW3 is in an open position; when low, SW3 is closed. Similarly, `ic_usb_sw4_en`, `ic_usb_sw5_en` and `ic_usb_sw6_en` control SW4, SW5, and SW6, respectively.

**Figure G-1 Resistor Control Positions for LS Mode of Operation**



#### G.1.4.1 Device Initial Attachment Sequence

The control of the switches mentioned above is illustrated in [Figure G-2](#).

**Figure G-2 IC\_USB Resistor Switch Control for LS Mode of Operation**

The HS OTG controller, at power-on reset, has the switch positions given in Sl. No. 1 of “[LS Mode Switch Position Matrix](#)” on page [749](#). The application of IC\_VDD from the host triggers the IC\_VDD monitor at the device to assert ic\_usb\_vbusvalid when IC\_VDD V<sub>OP</sub>. HS OTG Controller is then triggered so that it asserts ic\_usb\_sw5\_en (Sl. No. 2 of [Table G-1](#)). After a finite delay delay1 as shown in [Figure G-2](#), it de-asserts ic\_usb\_sw4\_en (Sl. No.3 of [Table G-1](#)). The device has now recognized the IC\_VDD and attached itself to the host. The host opens SW1 and SW2 in response (Sl. No. 4 of [Table G-1](#)). When the device detects reset, it opens SW6 by deasserting ic\_usb\_sw6\_en (Sl. No. 5 of [Table G-1](#)).

**Table G-1 LS Mode Switch Position Matrix**

Sl. No.	Condition	SW1	SW2	SW3	SW4	SW5	SW6	Comments
1	Power-on reset	closed	closed	closed	closed	open	closed	<a href="#">Fig. 6-4 of IC_USB ECN</a>
2	Device connect started	closed	closed	closed	closed	closed	closed	<a href="#">Fig. 6-5 of IC_USB ECN</a>
3	Device connect ended	closed	closed	closed	open	closed	closed	<a href="#">Fig. 6-5 of IC_USB ECN</a>
4	Device connect recognized by host	open	open	closed	open	closed	closed	<a href="#">Fig. 6-6 of IC_USB ECN</a>
5	Device recognizes reset	open	open	closed	open	closed	open	<a href="#">Fig. 6-7 of IC_USB ECN</a>
6	Device detach start	open	open	closed	closed	closed	open	<a href="#">Fig. 6-8 of IC_USB ECN</a>
7	Device detach end	open	open	closed	closed	open	open	<a href="#">Fig. 6-8 of IC_USB ECN</a>
8	Device attach start	open	open	closed	closed	closed	open	<a href="#">Fig. 6-9 of IC_USB ECN</a>
9	Device attach end	open	open	closed	open	closed	open	<a href="#">Fig. 6-9 of IC_USB ECN</a>

### G.1.4.2 Device Detach-Attach Sequence

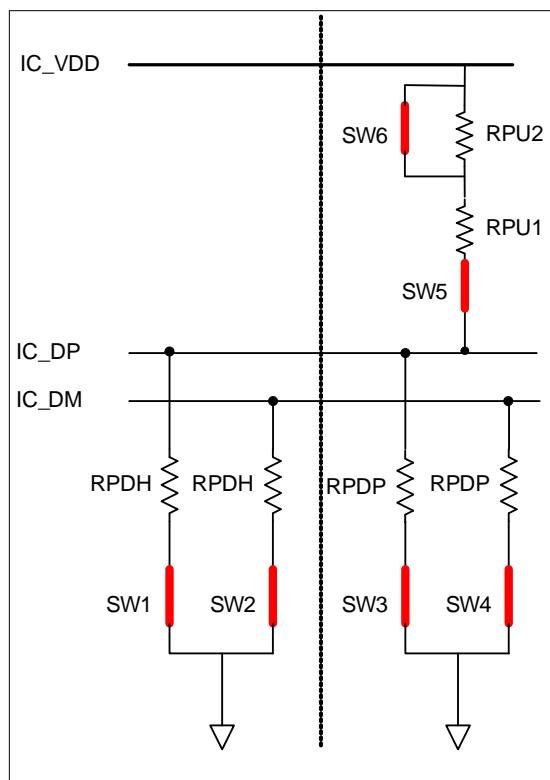
During the IC\_USB active operation, if a device detaches itself by performing a soft disconnect, it does so by closing SW4 by asserting `ic_usb_sw4_en` (Sl. No. 6 of [Table G-1](#) on page 749) followed by opening SW5 by de-asserting `ic_usb_sw5_en` (Sl. No. 7 of [Table G-1](#)) after a delay `delay2`. After detach, for re-attach, it closes SW5 by asserting `ic_usb_sw5_en` (Sl. No. 8 of [Table G-1](#)) followed by opening SW4 by de-asserting `ic_usb_sw4_en` (Sl. No. 9 of [Table G-1](#)) after a delay `delay3`.

### G.1.5 Control of SW3, SW4, SW5 and SW6 in FS Mode of Operation

[Figure G-3](#) shows the pull-up/pull-down resistor locations for FS mode of operation.

SW1 and SW2 are controlled by the host whereas SW3, SW4, SW5, and SW6 are controlled by the device.

**Figure G-3 Resistor Control Positions for FS Mode of Operation**



The HS OTG controller has individual controls for the 4 device controlled resistors.

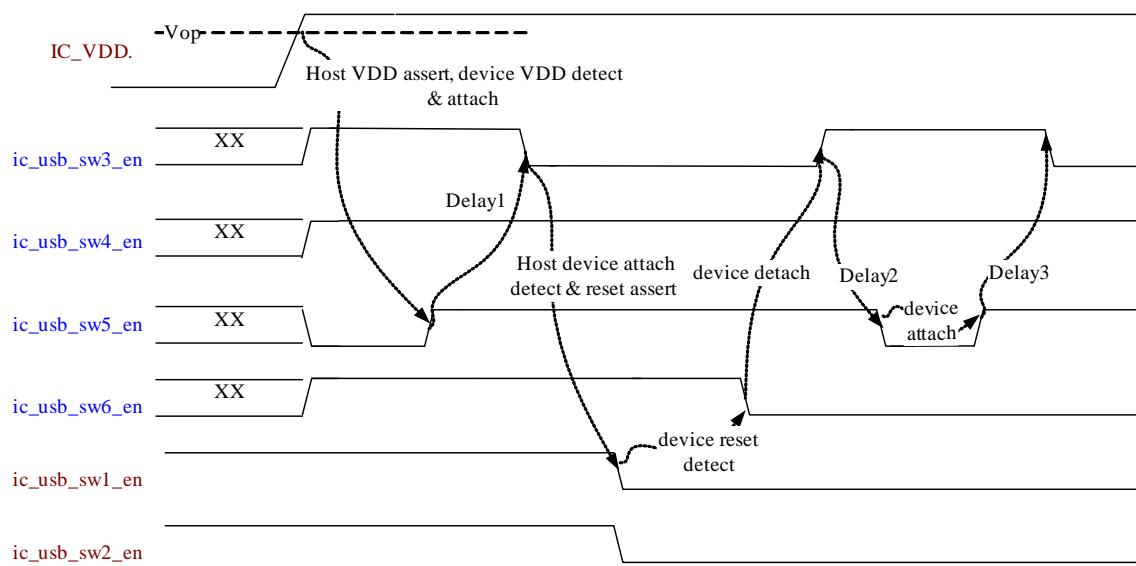
`ic_usb_sw3_en` is de-asserted (low) for SW3 to be in open position. `ic_usb_sw3_en` is asserted for SW3 to be in closed position.

Similarly, `ic_usb_sw4_en`, `ic_usb_sw5_en` and `ic_usb_sw6_en` control SW4, SW5 and SW6 respectively.

### G.1.5.1 Device Initial Attachment Sequence

The control of the switches mentioned is exactly that of LS mode, except the roles of SW3 and SW4 are reversed.

See [Figure G-4](#) for more detailed illustration.

**Figure G-4 IC\_USB Resistor Switch Control for FS Mode of Operation**

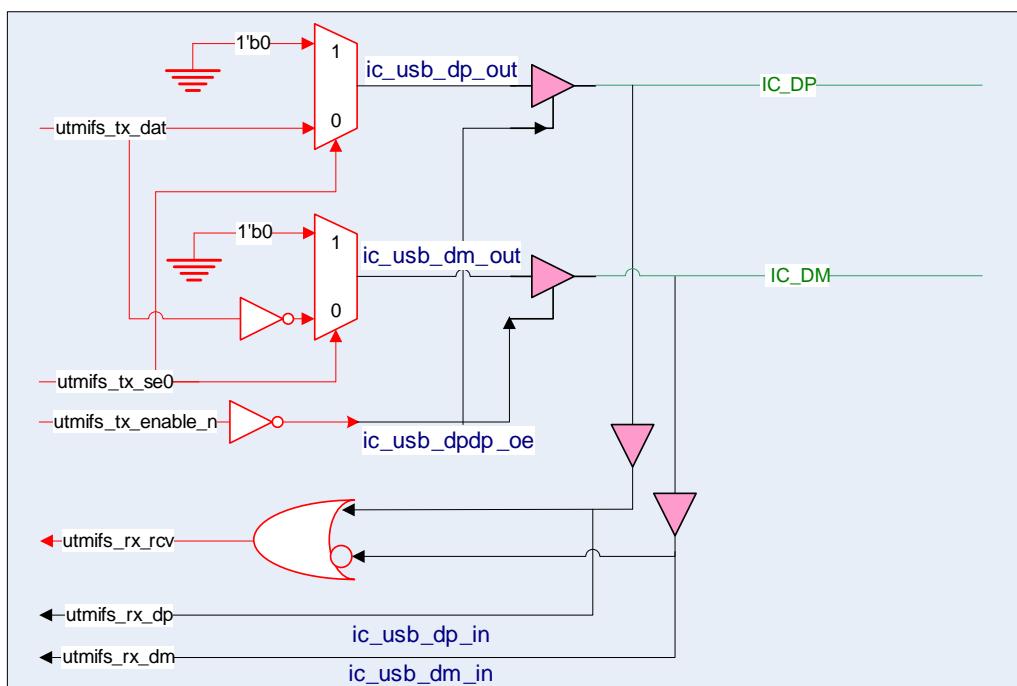
### G.1.5.2 Device Detach-Attach Sequence

The control of the switches mentioned is exactly similar to that of LS mode except with a major difference: The roles of SW3 and SW4 are reversed.

### G.1.6 Conversion of FS\_LS Signals to IC\_USB Signals

HS OTG currently already has a FS\_LS serial solution. IC\_USB signals are to be derived out of this solution using the logic shown in [Figure G-5](#).

The tri-state buffers, shown in pink, are outside the controller and must be implemented by the user of the controller.

**Figure G-5 FSLS Signals to IC\_USB Signals Conversion**

## G.2 Detailed Host-Only Functions

- Removable peripheral or fixed peripheral option
- Support for both self- or bus-powered peripheral
- Multiple voltage class
- Control of SW1 and SW2 in LS mode of operation
- Control of SW1 and SW2 in FS mode of operation
- Conversion of FS\_LS signals to IC\_USB signals

### G.2.1 Removable Peripheral or Fixed Peripheral Option

The host controller requires no additional functions to support this option.

### G.2.2 Support for Both Self- or Bus-Powered Peripheral

The host controller requires no additional functions to support this option.

### G.2.3 Multiple Voltage Class

To support multiple voltage class, external circuitry would be required to control the applied voltage level. No functions are performed within the controller. The external circuit needs to interact with the controller device driver software to find out the device attachment status with each step value of supplied voltage specified in section 6.2 of the IC\_USB spec. The controller provides the connection status of the port in its status register. The timers  $th_1$  and  $th_{new}$  defined in the specification need to be implemented in software for this purpose.

## G.2.4 Control of SW1 and SW2 in FS and LS Modes of Operation

See the device functions in section “[Control of SW3, SW4, SW5 and SW6 in LS Mode of Operation](#)” on page [748](#). The host has to detect the device initial attachment and de-assert `ic_usb_sw1_en` and `ic_usb_sw2_en`.

## G.2.5 Conversion of FS\_LS Signals to IC\_USB Signals

See the device functions in “[Conversion of FS\\_LS Signals to IC\\_USB Signals](#)” on page [751](#). The device and host functions for conversion from FS\_LS to IC\_USB are similar.

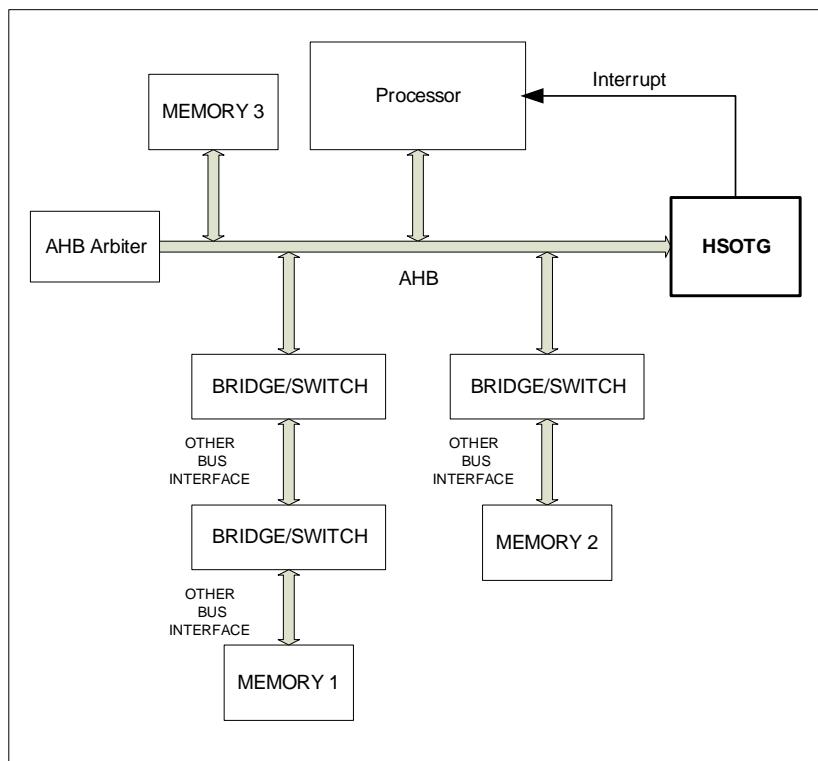


# H

## Remote Memory Support for Internal DMA Configurations

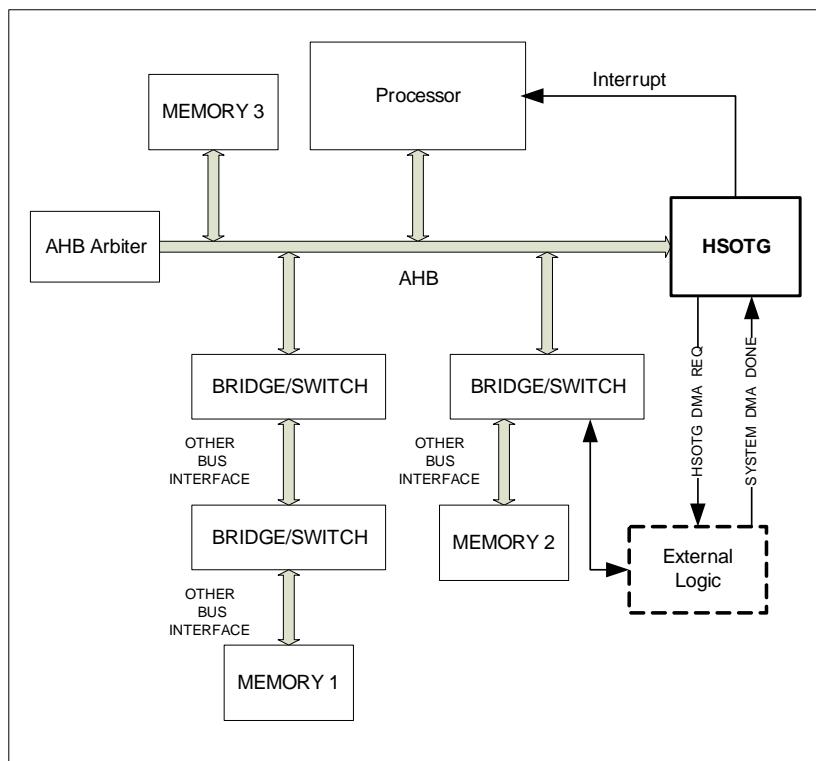
---

This appendix describes an HS OTG feature for a specific architectural scenario. In the internal DMA mode, after data is received on the USB interface, the HS OTG controller issues an XferComp interrupt immediately after the Last Data is written on the AHB bus to the external system memory. When the system architecture is designed to place memories in other clock domains through Bridges or Switches apart from the AHB interface, the XferComp interrupt might reach the processor even before all the relevant DATA reaches the external Memory. For example, in [Figure H-1](#), because the memory “MEMORY 2” is placed further down the system hierarchy, the DATA transfer to that memory might be delayed. In such a case, the HS OTG controller interrupts the processor before all the DATA has been completely written into Memory 2.

**Figure H-1 System Architecture with Memories in Different Hierarchies**

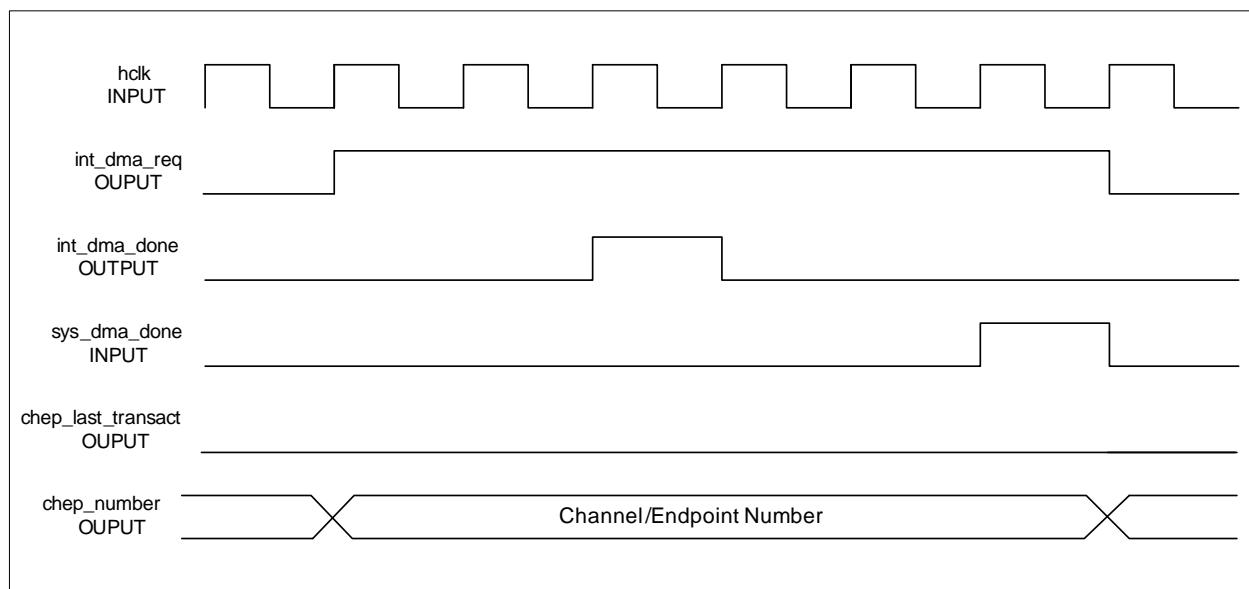
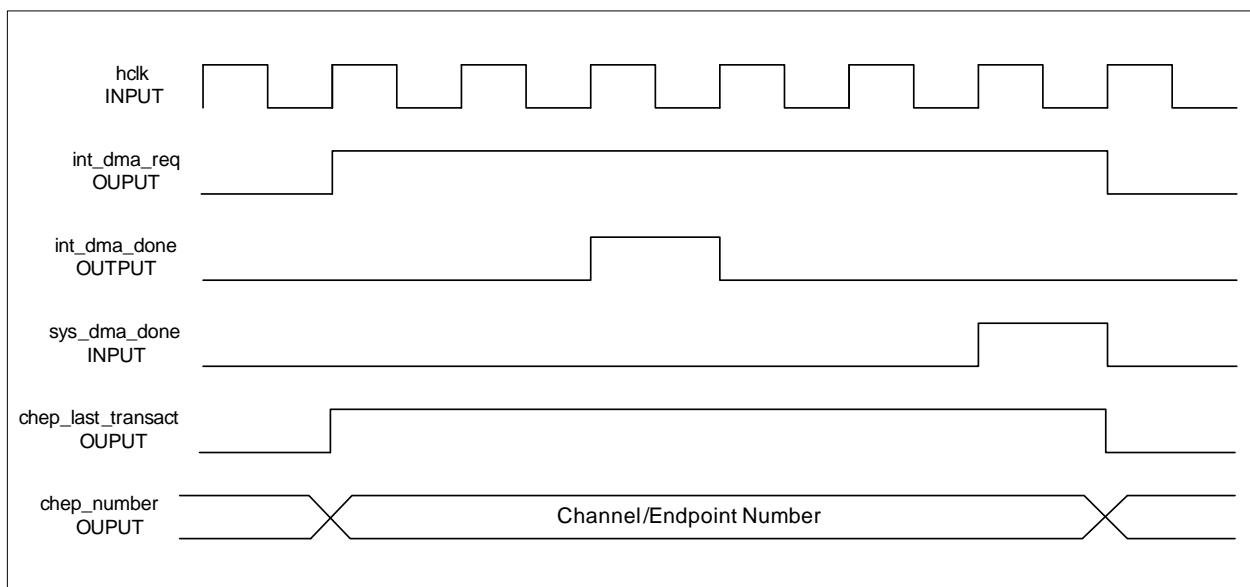
The Remote Memory Support functionality ensures that the HS OTG controller **waits till all the written DATA reaches the external Memory** before Xfer Comp interrupt is issued to the processor (see [Figure H-2](#)). The HS OTG controller waits for an external System DMA DONE signal to interrupt the system on Xfer Complete for the Data Transfer.

Note that the new feature is valid only for normal DATA transfer completion. When the transfer ends prematurely due to some fatal error conditions (STALL, XCS\_XACT\_ERR, Babble, and so on), the controller generates a halt interrupt without any DMA write requests.

**Figure H-2 DMA Remote Memory Support Functionality**

When GAHBCFG.RemMemSupp is set, the int\_dma\_req output signal is asserted when HS OTG DMA starts write transfer to the external memory. When the controller is done with the transfers, it asserts int\_dma\_done signal to flag the completion of DMA writes from HS OTG. The controller then waits for sys\_dma\_done signal from the system to proceed further and complete the Data Transfer corresponding to a particular Channel/Endpoint. The chep\_last\_transact and chep\_number signals are given as outputs carrying relevant information for the write transfers.

When GAHBCFG.RemMemSupp is zero (0), int\_dma\_req and int\_dma\_done signals are not asserted and the controller proceeds with the assertion of the XferComp interrupt as soon as the DMA write transfer is done at the HS OTG Controller Boundary and it doesn't wait for the sys\_dma\_done signal to complete the DATA transfers.

**Figure H-3 DMA RMS Signal Activity for All Except the Last Write Transaction****Figure H-4 DMA RMS Signal Activity for Last Write Transaction**

There are two modes of operation based on the NotiAllDmaWrit control bit in GAHBCFG register when GAHBCFG.RemMemSupp is set as described below:

1. GAHBCFG.NotiAllDmaWrit = 1,

HS OTG controller asserts int\_dma\_req for all the DMA write transactions including the last write transaction on the AHB interface along with int\_dma\_done, chep\_last\_transact and chep\_number signal informations.

The controller waits for sys\_dma\_done signal for every DMA write transaction including the Last Write transaction in order to complete the transfer of a particular Channel/Endpoint.

2. GAHBCFG.NotiAllDmaWrit = 0,

HS OTG controller asserts int\_dma\_req signal only for the last transaction of DMA write transfer corresponding to a particular Channel/Endpoint.

Similarly, the controller waits for sys\_dma\_done signal only for that transaction of DMA write to complete the transfer of a particular Channel/Endpoint.

## H.1 Last Transaction in Remote Memory Support

- Buffer DMA Mode – The Last DMA Transaction for a write transfer is when the received packet is written to the system memory and the current Endpoint's or Channel's Xfersize is equal to its MPS or when the received packet is a short packet or a zero length packet.
- Descriptor DMA Mode – The Last DMA Transaction for a write transfer is when the received packet is written to the system memory and any one of the below conditions is met.
  - IOC bit is set for the descriptor entry.
  - MTRF bit is not set for the descriptor entry and the received packet was a short packet.

Special Cases when it is not the Last Transaction:

- In device mode, when a short packet is received and MTRF=0 for an ISOC OUT transfer, last transaction is not generated as the controller continues without an interrupt.
- In device mode, when there is a Descriptor Update for an IN transfer and Short Packet bit is set with MTRF=0.

## H.2 Requirements

The setup requires the system DMA DONE (sys\_dma\_done) signal to be set for one AHB clock cycle immediately after all the relevant DATA is written to the memory.

## H.3 Mode of Operation

The functionality is valid only for internal DMA write transfers to the External Memory. It applies to the following cases:

Buffer DMA Mode (Non-Descriptor internal DMA):

- IN Transaction in Host Mode
- OUT Transaction in Device Mode

### Scatter Gather DMA Mode (Descriptor internal DMA) -

- IN Data Buffer Write Transaction in Host Mode.
- IN/OUT Descriptor Update Write Transaction in Host Mode.
- OUT Data Buffer Write Transaction in Device Mode.
- IN/OUT Descriptor Update Write Transaction in Device Mode.



# DWC\_otg Hibernation Add-on Feature



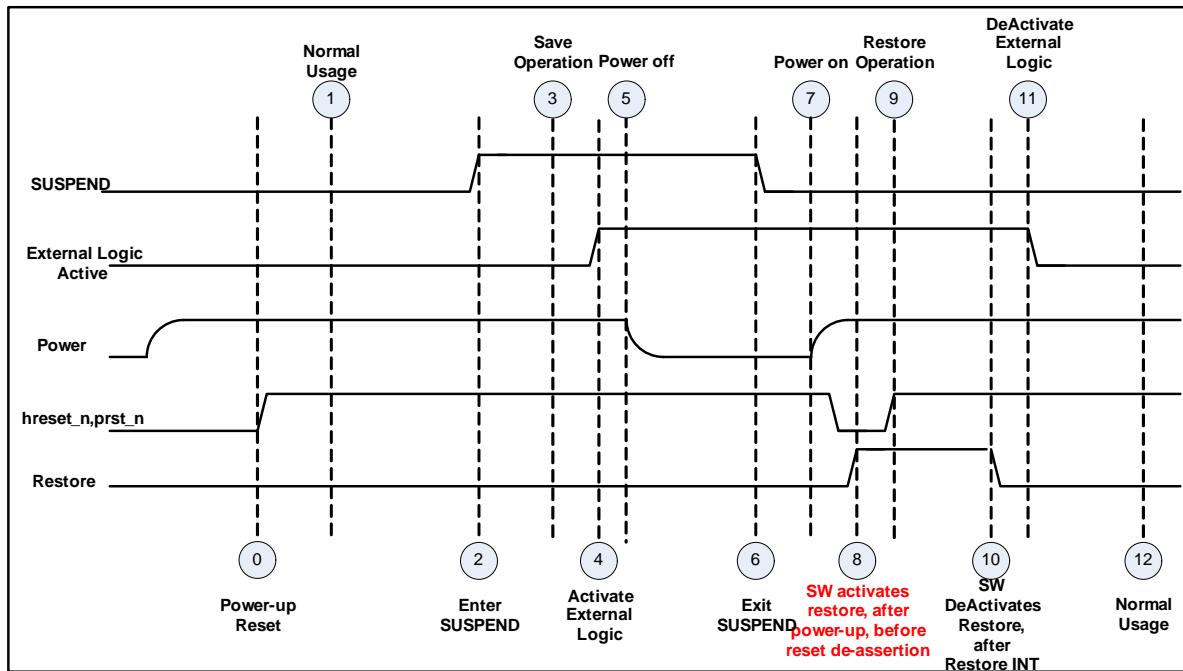
This is an add-on feature that requires an additional DWC-HSOTG-HIBERNATION license.

This appendix describes hibernation support for the HS OTG controller.

## I.1 Overview of the Hibernation Sequence

This section describes the sequence of events that occur between the controller and its application for hibernation functionality. [Figure I-1](#) illustrates this sequence. Map the numbers listed in the diagram to the event number in [Table I-1](#) on page 762.

**Figure I-1** Hibernation Sequence of Events



**Table I-1 Hibernation Sequence of Events**

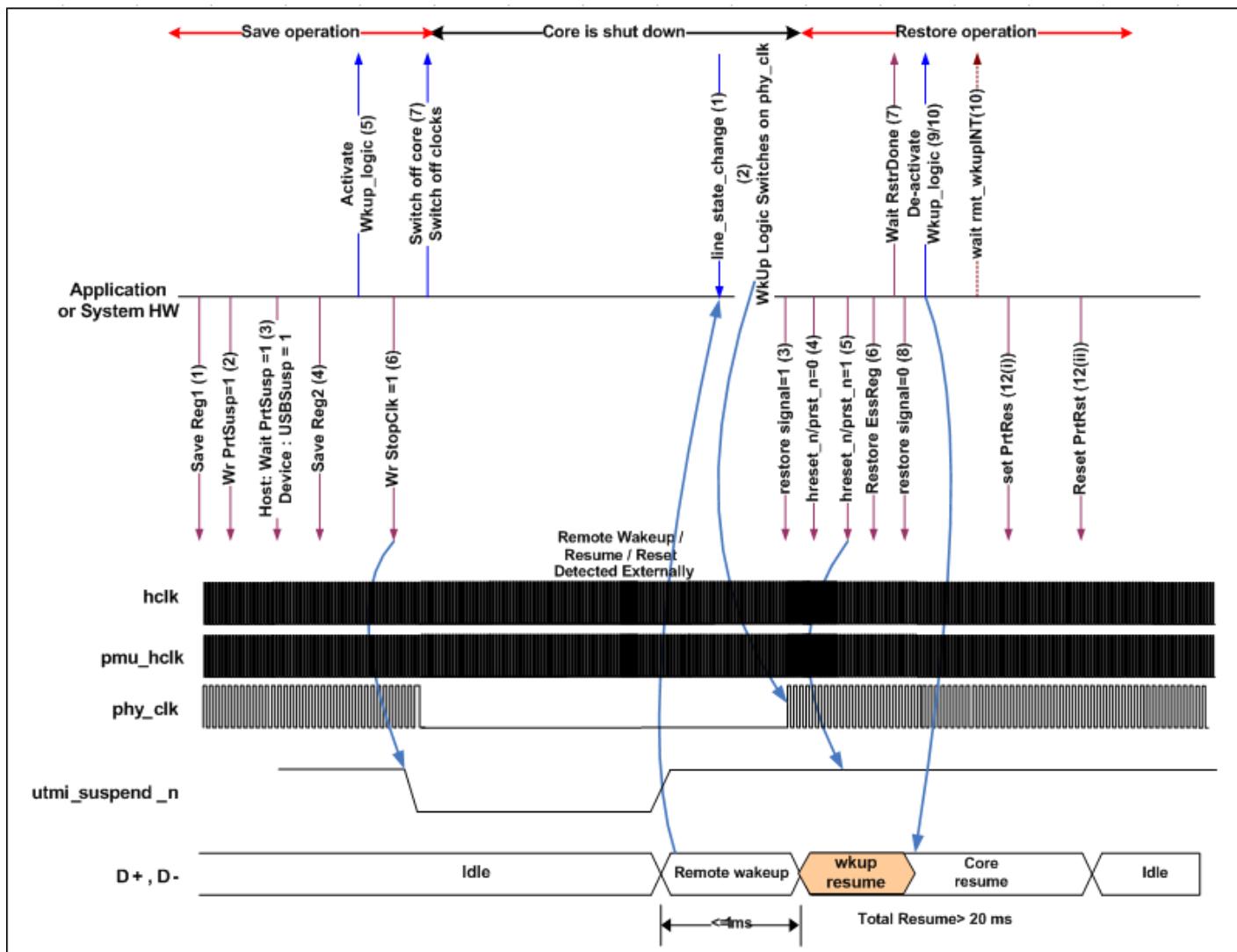
Event #	Description
0	The power on reset to the controller is de-asserted.
1	The controller (acting as either as host or device) starts normal operations
2	If the controller is in host mode, USB SUSPEND is initiated. If the controller is in device mode, USB SUSPEND is detected.
3	The application prepares to put the controller into hibernation by initiating a “SAVE” operation on the essential registers and states.
4	The external logic, which has the primary responsibility of detecting and informing the application about any activity on the bus when the controller is in hibernation, is activated.
5	The controller (apart from the external logic) is powered off.
6	The controller exits SUSPEND when it detects activity on the bus or if it wants to initiate activity on the bus.
7	The controller is powered on.
8	The application activates the “RESTORE” signal to the controller indicating that it is coming out of hibernation and needs to restore the essential registers and states.
9	The application restores essential registers and states.
10	The application completes the “RESTORE” by deactivating the restore signal to the controller.
11	The external logic is de-activated.
12	The controller (acting as either host or device) resumes normal operations.

### I.1.1 DWC\_otg Controller Functionality for Hibernation

The hibernation functionality can be described by segmenting it into two operations - the SAVE operation and the RESTORE operation

The SAVE operation occurs at the entry of SUSPEND state (step 2 in [Figure I-1](#)). The controller stays inactive on the USB bus for 3 msec indicating that the controller has entered Suspend state. The application ascertains that the controller is in Suspend state by reading HPRT.PrtSusp in host mode and GINTSTS.USBSusp in device mode. The application then transfers RESTORE information read from controller CSR into its system memory. The external wakeup detection logic is activated by asserting wkup\_active. Then the controller is switched off. This time instance is defined as SAVE\_POINT.

See [Figure I-2](#) for a diagrammatic representation of the process.

**Figure I-2** The Hibernation SAVE and RESTORE Operations

In [Figure I-2](#), the time instance just before controller is powered off is defined as the `SAVE_POINT`. The time instance at which controller is powered back is defined as `RESTORE_POINT`.



For Host Mode ULPI configuration, the application must program the `GUSBCFG.ULPIAutoRes` bit before entering suspend if Hibernation feature is to be used and expects a remote wakeup from the device.

## I.1.2 Signal Changes to Support Hibernation

All the signals from the controller required for the functionality of the wakeup logic are bundled together in a bus called core\_gen\_out. [Table I-2](#) lists and describes the signals that are taken out of the DWC\_otg\_core.

**Table I-2** Signal description for gen\_out interface of the DWC\_otg\_core

Bit	Signal Name	Mode	Functionality
11:10	HCFG/DCFG Speed	Host and Device	<p>Indicates the speed at which the application requires the controller to enumerate, or the maximum speed the application can support.</p> <ul style="list-style-type: none"> <li>■ 2'b00: High speed (USB 2.0 PHY clock is 30 MHz or 60 MHz)</li> <li>■ 2'b01: Full speed (USB 2.0 PHY clock is 30 MHz or 60 MHz)</li> <li>■ 2'b10: Reserved</li> <li>■ 2'b11: Full speed (USB 1.1 transceiver clock is 48 MHz)</li> </ul>
9	OtgVersion	Host and Device	<p>Indicates whether OTG Revision 1.3 or OTG Revision 2.0 is supported</p> <ul style="list-style-type: none"> <li>■ 1'b0: OTG 1.3 is supported</li> <li>■ 1'b1: OTG 2.0 is supported</li> </ul>
8	GUSBCFG.IC_USBCap	Host and Device	<p>The application uses this bit to control the DWC_otg controller's IC_USB capabilities. If the controller operates as non-IC_USB-capable, it can only connect to non- IC_USB-capable PHYs. If the controller operates as IC_USB-capable, it can only connect to IC_USB-capable PHYs.</p> <ul style="list-style-type: none"> <li>■ 1'b0: IC_USB capability is not enabled.</li> <li>■ 1'b1: IC_USB capability is enabled.</li> </ul> <p>This bit is writable only if:</p> <ul style="list-style-type: none"> <li>■ An IC_USB mode was specified for Mode of Operation in coreConsultant (parameter OTG_ENABLE_IC_USB) and</li> <li>■ IC_USB_MODE and FS_LS feature are both enabled simultaneously. The reset value is 1'b0 when the FS_LS feature is enabled. Otherwise, this bit is set to 1'b0 and the bit is read-only.</li> </ul>
7	USBCFG.ULPI_UTMI_SEL	Host and Device	<p>The application uses this bit to select either a UTMI+ interface or ULPI Interface.</p> <ul style="list-style-type: none"> <li>■ 1'b0: UTMI+ Interface</li> <li>■ 1'b1: ULPI Interface</li> </ul> <p>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>

**Table I-2 Signal description for gen\_out interface of the DWC\_otg\_core**

<b>Bit</b>	<b>Signal Name</b>	<b>Mode</b>	<b>Functionality</b>
6	GUSBCFG.PHYIf	Host and Device	<p>The application uses this bit to configure the controller 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.</p> <ul style="list-style-type: none"> <li>■ 1'b0: 8 bits</li> <li>■ 1'b1: 16 bits</li> </ul> <p>This bit is writable only if UTMI+ and ULPI were selected in coreConsultant configuration (parameter = 2). Otherwise, this bit returns the value for the power-on interface selected during configuration.</p>
5:4	DCFG.DevSpd / HPRT.PrtSpd	Host and Device	<p>This bit indicates the speed at which the application requires the controller 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 controller is connected.</p> <ul style="list-style-type: none"> <li>■ 2'b00: High speed (USB 2.0 PHY clock is 30 MHz or 60 MHz)</li> <li>■ 2'b01: Full speed (USB 2.0 PHY clock is 30 MHz or 60 MHz)</li> <li>■ 2'b10: Reserved</li> <li>■ 2'b11: Full speed (USB 1.1 transceiver clock is 48 MHz)</li> </ul>
3	If_dev_mode	Host and Device	<p>This bit indicates whether the controller is functional as a device or as a host</p> <ul style="list-style-type: none"> <li>■ 1'b0: Indicates that the controller is functional as a USB Host</li> <li>■ 1'b1: Indicates that the controller is functional as a device</li> </ul>
2	GUSBCFG.PHYSel	Host and Device	<p>The application uses this bit to select either a high-speed UTMI+ or ULPI PHY, or a full-speed transceiver.</p> <ul style="list-style-type: none"> <li>■ 1'b0: USB 2.0 high-speed UTMI+ or ULPI PHY</li> <li>■ 1'b1: USB 1.1 full-speed serial transceiver</li> </ul> <p>If a USB Full-Speed Serial Transceiver interface is not selected in coreConsultant (parameter OTG_FSPHY_INTERFACE = 0), this bit is always 0, with write-only access.</p> <p>If a high-speed PHY interface is not selected in coreConsultant (parameter OTG_HSPHY_INTERFACE = 0), this bit is always 1, with write-only access.</p> <p>If both interface types are selected in coreConsultant (parameters have non-zero values), the application uses this bit to select the active interface. Access is read and write.</p>

**Table I-2** Signal description for gen\_out interface of the DWC\_otg\_core

Bit	Signal Name	Mode	Functionality
1	GUSBCF.FSIntf	Host and Device	<p>The application uses this bit to select either a unidirectional or bidirectional USB 1.1 full-speed serial transceiver interface.</p> <ul style="list-style-type: none"> <li>■ 1'b0: 6-pin unidirectional full-speed serial interface</li> <li>■ 1'b1: 3-pin bidirectional full-speed serial interface</li> </ul> <p>If a USB 1.1 Full-Speed Serial Transceiver interface is not selected in coreConsultant (parameter OTG_FSPHY_INTERFACE = 0), this bit is always 0, with write-only access.</p> <p>If a USB 1.1 FS interface is selected in coreConsultant (parameter OTG_FSPHY_INTERFACE! = 0), then the application can set this bit to select between the 3- and 6-pin interfaces. Access is read and write.</p>
0	GINTSTS.RestoreDone	Host and Device	This is the restore done interrupt status bit, the assertion of this bit indicates to the wakeup logic and the power off slave that the controller has been restored completely and the wakeup logic can now be switched off.



**Note** All the signals listed in [Table I-2](#) are clamped and latched in the PMU module once the clamp logic is activated by the application (GPWRDN.PMUActv = 1'b1).

### I.1.3 Scenarios Illustrating DWC\_otg Controller Behavior When PHY Clock is On when Controller is in Off State (PMU Active)

#### I.1.3.1 Host Mode

##### Scenario 1

When the Host VBUS is ON (HPRT.PrtPwr = 1) but no device is connected, the application programs the PCGCCTL.StopPclk to stop the PHY clock. When the controller is switched off to save power and enable the PMU, the PHY clock will not be OFF.

##### Scenario 2

When the Host controller VBUS is ON, a device is connected and the USB is in Suspend state, the application programs PCGCCTL.StopPclk to switch off the PHY clock. The application then switches off the controller to save power and enable the PMU. In the controller-off state, if the device is disconnected the PHY clock will be switched back ON. If the application decides to maintain the controller in OFF state, the PHY clock will still be ON and also the VBUS.

In both the above scenarios, the next steps in the USB flow could be:

1. Dev Connect
2. Controller is switched back on by the application

3. Cable is reversed leading to ID change

## Solutions for Non-ULPI Interfaces

There are two possible solutions for non-ULPI interfaces:

- Solution 1: After enabling PMU, the application can program the GPWRDN.VbusOff register to enable switching off the PHY Clock and also the VBUS.
- Solution 2: Before Enabling PMU, the application can program the HPRT.PrtPwr to 1'b0 and then after enabling PMU, program the GPWRDN.VbusOff register to switch off the PHY clock.

## Solution for ULPI Interfaces

Before Enabling PMU, the application can program the HPRT.PrtPwr to 1'b0 and then after enabling PMU, program the GPWRDN.VbusOff register to switch off the PHY clock.

In this solution, the Device connect is not detected because the VBUS is OFF (unless there is an SRP from the Device). If the Host controller is not SRP capable, then the application should periodically switch ON the controller and then program HPRT.PrtPwr to switch ON the VBUS and check for any device connect.

### I.1.3.2 Device Mode

When the Device controller is in Suspend state, the application programs PCGCCTL.StopPclk to switch off the PHY clock. The application then switches off the controller to save power and enable the PMU. In the controller-off state, if the VBUS is switched OFF (Session End) the PHY clock will be switched back ON. If the application decides to maintain the controller in OFF state, the PHY clock and the VBUS will remain ON.

The next steps in the USB flow could be:

1. VBUS ON (Session Valid from Host)
2. Application switches on the controller (to perform SRP)
3. Cable is reversed leading to ID change.

## Possible Solution for Non-ULPI Interfaces

The application can program the GPWRDN.VbusOff register to enable switching off the PHY Clock.

## Possible Solution for ULPI Interfaces

The application must switch ON the controller, then program the PCGCCTL.StopPclk to 1'b1 to switch off the PHY clock. The application must then switch off the controller and enable the PMU.



# J

## UTMI-to-UTMI Bridge Add-on Feature



This Add-On Feature is no longer supported from v4.20a. Contact Synopsys IP support if you are using this feature currently or intending to do so for a new project.

This appendix describes the UTMI-to-UTMI bridge. The terms “U2U Bridge,” “DWC\_U2UB controller,” and “DWC\_U2U Bridge controller” are used interchangeably. The following topics are discussed in this appendix:

- “Overview of U2U Bridge” on page 770
- “U2U Bridge Operational Sequences” on page 776
- “U2U Bridge Clock Logic” on page 791
- “U2U Bridge Reset Logic” on page 793
- “Configuration Parameter” on page 796
- “Signal Descriptions” on page 797
- “Registers” on page 827
- “Area” on page 827
- “References” on page 828
- “Integrating DWC\_U2UB Device Interface with DWC\_otg Controller in Non-OTG Device Mode” on page 829



- The DWC\_U2UB controller is available only as a licensable feature as a part of the DWC\_otg controller. To enable this feature, you are required to have a DWC-USB-U2UB license.
- The DWC\_U2UB controller can be used with the DWC\_otg controller only if the following configuration requirements are met. The DWC\_otg must be configured as:
  - Device Only or Host Only (OTG\_MODE = 3, 4, 5 or 6)
  - UTMI only interface (OTG\_HSPHY\_INTERFACE = 1)
  - Either in 8-bit UTMI Data width or in 16-bit UTMI Data width (OTG\_HSPHY\_WIDTH = 1 or 2)

## J.1 Overview of U2U Bridge

This section discusses the following topics:

- “[Introduction](#)” on page [770](#)
- “[U2U Bridge Functions](#)” on page [770](#)
- “[Controller Requirements](#)” on page [772](#)
- “[U2U Bridge - System-Level Block Diagram](#)” on page [773](#)

### J.1.1 Introduction

Driven by a need for eliminating analog components in USB 2.0 protocol and saving power further, UTMI-to-UTMI (U2U) Bridge core provides an alternate method to connect a device controller to a host controller, when both controllers support UTMI interface. Instead of connecting the UTMI interfaces exposed by the device and host controllers to a standard UTMI PHY, the interfaces are connected to the U2U Bridge. The U2U Bridge performs functions necessary to successfully connect (attach) the device to the host and perform protocol related-functions like enumeration, exchange of USB traffic, suspend/resume and disconnect.

### J.1.2 U2U Bridge Functions

The U2U Bridge performs the following functions:

- Emulates PHY functionality at the device and host UTMI interface
- Informs about the following host events to device:
  - Presence of VBUS
  - Host Reset
  - Host-initiated L1 Suspend
  - Host-initiated L2 Suspend
  - Host-initiated resume (L1 & L2)
  - Absence of VBUS (Host disconnect)
- Informs about the following device events to host:
  - Device Connect
  - Device-initiated remote wakeup (L1 & L2)
  - Device disconnect
- Manages USB traffic exchange between host controller and device controller using the 8-bit/16-bit UTMI datapath
- Manages provisioning and gating of 60-MHz UTMI clock (or 30-MHz UTMI clock in case of 16-bit UTMI datapath) to host and device controllers
- Handles the previously-mentioned functionalities by low gate count logic external to the device/host controller cores. The logic used for detecting the events on either side of the bridge is compatible to 8-bit/16-bit UTMI interface and additionally as defined in the functionality described in this appendix.

### J.1.2.1 U2U Bridge Supported Usage Modes

[Table J-2](#) on page [774](#) lists the supported usage modes of U2U Bridge.

**Table J-1 U2U Bridge Usage Modes**

Category	Support
Speed	<ul style="list-style-type: none"> <li>■ Full Speed (FS) only</li> <li>■ No support for High Speed (HS) and Low Speed (LS)</li> </ul>
Mode of operation	<ul style="list-style-type: none"> <li>■ Non-OTG Device</li> <li>■ Non-OTG Host</li> <li>■ SRP-Capable Host</li> <li>■ SRP-Capable Device</li> </ul> <p>No support for Dual-Role Device (DRD) Modes.</p>
Interface type	<p>8-bit/16-bit UTMI+ with FS dedicated interface applicable only for host mode during suspend.</p> <ul style="list-style-type: none"> <li>■ The parallel UTMI+ interface is used for:           <ul style="list-style-type: none"> <li>- All normal UTMI+ RX/TX packets</li> <li>- Suspend/Resume on device and host side</li> <li>- USB Reset on host side</li> </ul> </li> <li>■ The serial UTMI interface can be used for Suspended (L1 &amp; L2) port Resume on host side</li> </ul>
Clocks	<ul style="list-style-type: none"> <li>■ The UTMI clocks (nominally 60MHz), with accuracy and duty cycle defined as per the UTMI Specification (<math>\pm 500</math> ppm, <math>50\% \pm 10\%</math> duty cycle) on the device UTMI interface and host UTMI interface are asynchronous to each other.</li> <li>■ The U2U Bridge needs an auxiliary clock (auxclk) at a nominal frequency of 24MHz (with <math>\pm 2000</math> ppm accuracy and <math>50\% \pm 5\%</math> duty cycle) during suspend, resume, and remote wakeup. It switches off the UTMI clocks during this period.</li> </ul>

### J.1.2.2 U2U Bridge Unsupported Features

The following features are excluded from the scope of the product:

- Partial or complete power shut down during normal operation of the bridge including suspend/resume scenarios
  - Power clamping internal to the U2U Bridge core
  - High speed support
- The USB speed of the controllers using the U2U Bridge must be full speed (12 Mbps). No support for high speed or low speed.
- Scalability: The U2U Bridge is not scalable for multiple ports. It is intended to be used for single port. The UTMI clock frequencies need to follow the *UTMI Specification* and cannot be scaled.

- No synchronous reset support: The USB controllers using the U2U Bridge must use Asynchronous reset.



**Note** You must handle the MUX selection of the external MUX, that selects HSPHY and U2U Bridge combination, in the system. This is not the scope of the U2U Bridge controller.

## J.1.3 Controller Requirements

This section lists the device and host controller requirements to use the U2U Bridge.

### J.1.3.1 Device Controller Requirements

The device controller must meet the following requirements to use the U2U Bridge:

- Work only as a full-speed device, even when connected to a high-speed host
- Must not respond with HS Chirp during host HS reset
- Use the parallel UTMI+ interface for suspend/resume on device side
- Must not expect UTMI clock to be switched on in suspend state

### J.1.3.2 Host Controller Requirements

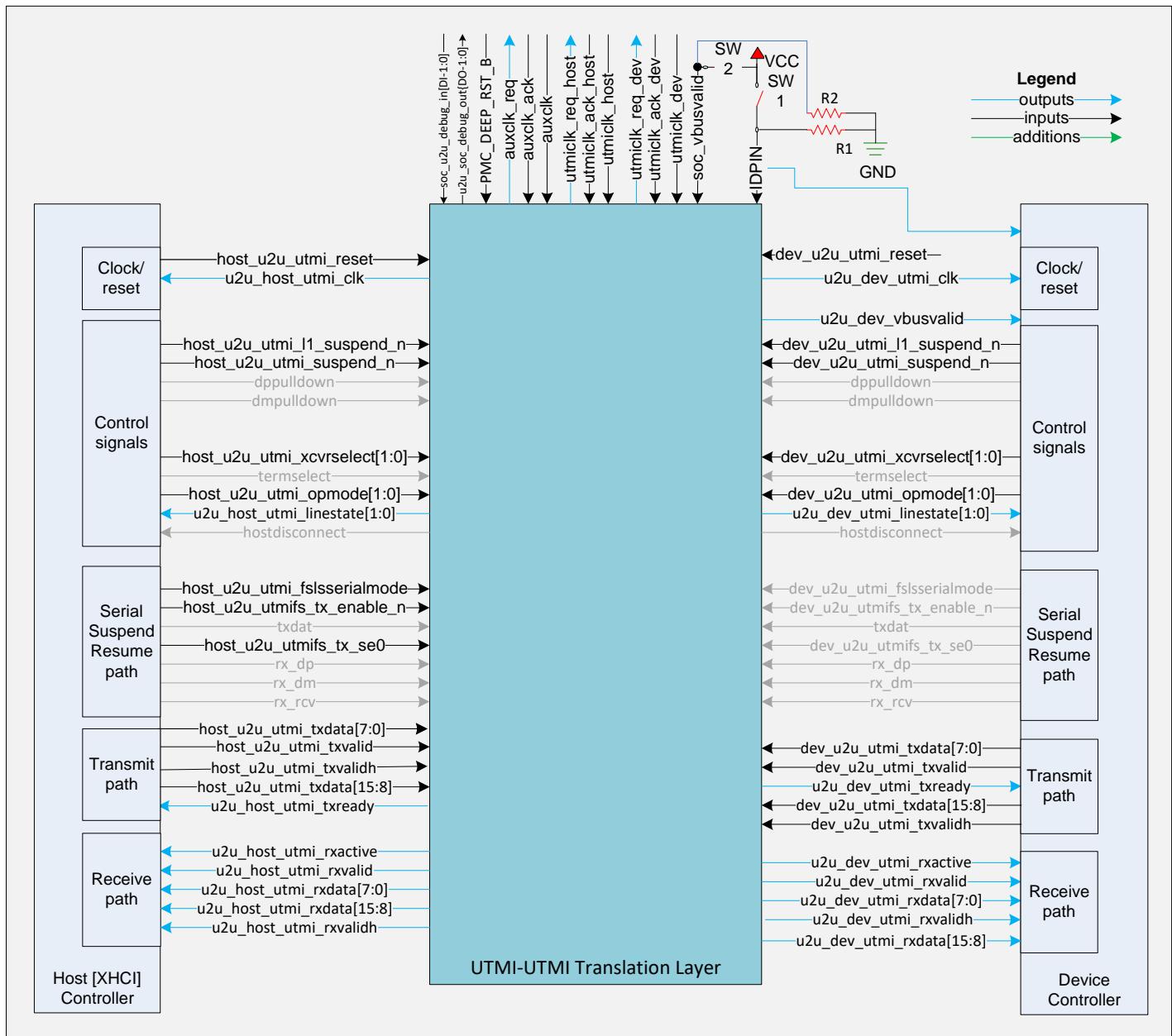
The host controller must meet the following requirements to use the U2U Bridge:

- Use the serial UTMI+ interface for Suspended (L1 & L2) port Resume on host side
- Must not expect UTMI clock to be switched on in Suspend state

## J.1.4 U2U Bridge – System-Level Block Diagram

Figure J-1 shows the system-level block diagram of the U2U Bridge.

**Figure J-1 U2U Bridge – System-Level Block Diagram**



**Table J-2** lists the UTMI+ Level 0 and Level 1 signals that are not used or driven by the U2U Bridge.

**Table J-2 UTMI+ Level 0/1 Signals not Used/Driven by U2U Bridge**

To/From Host/Device Controller	Signals Unused by U2U Bridge	Signals Not Driven by U2U Bridge
UTMI+ Level 0 Signals		
Host	<ul style="list-style-type: none"> <li>■ utmi_termselect</li> <li>■ utmi_databus16_8</li> </ul>	<ul style="list-style-type: none"> <li>■ utmi_rxerror</li> </ul>
Device	<ul style="list-style-type: none"> <li>■ utmi_termselect</li> <li>■ utmi_databus16_8</li> </ul>	<ul style="list-style-type: none"> <li>■ utmi_rxerror</li> </ul>
UTMI+ Level 1 Signals		
Host	<ul style="list-style-type: none"> <li>■ utmi_dppulldown, utmi_dmpulldown, utmi_dppullup, utmi_dmpullup</li> <li>■ utmifs_txdat</li> <li>■ utmifs_rx_dp, utmifs_rx_dm, utmifs_rx_rcv</li> <li>■ IDPULLUP, drrvbus, chrgvbus, dischrgvbus, txbitstuffenable, txbitstuffenableh</li> </ul>	<ul style="list-style-type: none"> <li>■ Hostdisconnect, avalid, bvalid, sessend</li> <li>■ Vbusvalid</li> </ul>
Device	<ul style="list-style-type: none"> <li>■ utmi_dppulldown, utmi_dmpulldown, utmi_dppullup, utmi_dmpullup</li> <li>■ utmifs_txdat</li> <li>■ utmifs_rx_dp, utmifs_rx_dm, utmifs_rx_rcv</li> <li>■ IDPULLUP, drrvbus, chrgvbus, dischrgvbus, txbitstuffenable, txbitstuffenableh</li> <li>■ utmi_fslsserialmode, utmifs_tx_enable_n, utmifs_tx_se0</li> </ul>	<ul style="list-style-type: none"> <li>■ Hostdisconnect, avalid, bvalid, sessend</li> </ul>

When 16-bit datapath is enabled, the U2U Bridge,

- Uses the following UTMI+ Level 0 signals from both device and host controllers:
  - utmi\_txdata[15:8]
  - utmi\_txvalidh
- Drives the following UTMI+ Level 0 signals to both device and host controller:
  - utmi\_rxdata[15:8]
  - utmi\_rxvalidh.

The U2U Bridge does not use IDPIN signal.

[Table J-3](#) lists the RTL design files of the U2U Bridge.

**Table J-3 DWC\_U2U Bridge – RTL Design Files**

Design Files	Description
DWC_U2UB_params.v	DWC_U2UB parameter defines
DWC_U2UB_derived_params.v	DWC_U2UB derived parameters
DWC_U2UB.V	Top-level design
DWC_U2UB_dev_ctrl.v	U2UB Device Controller Interface, generates device line state output
DWC_U2UB_bcm21.v	Double-stage synchronizer
DWC_U2UB_bcm22.v	Pulse synchronizer
DWC_U2UB_bcm74.v	Dual Clock FIFO Controller
DWC_U2UB_host_ctrl.v	DWC_U2UB Host Controller interfaces, generates host line state output
DWC_U2UB_bcm21.v	Double-stage synchronizer
DWC_U2UB_bcm22.v	Pulse synchronizer
DWC_U2UB_bcm74.v	Dual Clock FIFO Controller
DWC_U2UB_clk_req.v	Generates the clock request
DWC_U2UB_bcm21.v	Double-stage synchronizer
DWC_U2UB_pkt_emulator.v	Generates the linestate output during sync & EOP for TX/RX
DWC_U2UB_clk_mux.v	Clock multiplexer
DWC_U2UB_clkgate_cell.v	Clock gate cell
DWC_U2UB_clk_or.v	"OR" cell used for clock gating
DWC_U2UB_bussync.v	Bus Synchronizer
DWC_U2UB_mux.v	Multiplexer
DWC_U2UB_sync_ctl.v	Multi-Bit Double/Toggle Synchronizer
DWC_U2UB_clamp.v	Power clamp

## J.2 U2U Bridge Operational Sequences

This section discusses the following topics:

- “[Connect Sequence](#)” on page [776](#)
- “[Host Reset Sequence Following Initial Device Connect](#)” on page [778](#)
- “[Host Reset Sequence During Normal Traffic](#)” on page [779](#)
- “[Normal Traffic Flow in FS Idle Following Host Reset](#)” on page [779](#)
- “[Suspend Sequence](#)” on page [781](#)
- “[Device-Initiated Remote Wakeup Sequence](#)” on page [783](#)
- “[Host-Initiated Resume Sequence](#)” on page [785](#)
- “[Host-Initiated Resume Sequence in Parallel Mode](#)” on page [787](#)
- “[Host Reset During Suspend Sequence](#)” on page [788](#)
- “[Disconnect Sequence](#)” on page [790](#)

### J.2.1 Connect Sequence

The device attach sequence on the device side is as follows:

1. The U2U Bridge is driving `u2u_host_utmi_linestate` to SE0 state (`2'b00`).
2. The U2U Bridge waits for `soc_vbusvalid` to be asserted. The U2U Bridge drives `u2u_host_utmi_linestate` and `u2u_dev_utmi_linestate` to J state (`2'b01`) provided `soc_vbusvalid` is asserted. When `soc_vbusvalid` is de-asserted, the U2U bridge must continue to maintain `u2u_host_utmi_linestate` and `u2u_dev_utmi_linestate` to SE0 irrespective of the opmode from the device or host controllers.
3. SoC combinatorially de-asserts `dev_u2u_utmi_reset`, and the device controller combinatorially de-asserts `dev_u2u_utmi_suspend_n` and `dev_u2u_utmi_l1_suspend_n` to the U2U Bridge allowing the U2U Bridge to assert `utmiclk_req_dev` request.
4. SoC switches on `utmiclk_dev`, and asserts `utmiclk_ack_dev`.
5. The device controller is fed with `u2u_dev_utmi_clk`. A clock gate is present within the U2U Bridge to block this clock in L1 and L2 suspend.
6. The device controller may optionally monitor for `u2u_dev_vbusvalid =1` before it initiates the connect sequence. If the device controller does not monitor these signals before connect sequence, the device controller will continue to see SE0 state on `u2u_dev_utmi_linestate` from the U2U bridge. The device controller will infer that the host is not connected, because it expects `u2u_dev_utmi_linestate` to be in J (`2'b01`) state followed by reset sequence (SE0, back to J,...) after it initiates the connect sequence.
7. The device controller drives `dev_u2u_utmi_xcvrselect` to `2'b01` (FS mode) and `dev_u2u_utmi_opmode` to `2'b00` within the time  $T_{SIGATT}$  (Refer to the *UTMI Specification* ( $T_{SIGATT} < 100\text{ ms}$ )) to initiate the connect sequence and waits for `u2u_dev_utmi_linestate` to change from SE0 to J and back to SE0 (`2'b00`) – Host reset to be asserted.

The U2U Bridge does not take any action based on  $T_{SIGATT}$ .

The device attach sequence on the host side is as follows:

1. The U2U Bridge waits for `soc_vbusvalid` to be asserted. The U2U Bridge drives `u2u_host_utmi_linestate` and `u2u_dev_utmi_linestate` to J state (2'b01) provided `soc_vbusvalid` is asserted. When `soc_vbusvalid` is de-asserted, the U2U bridge continues to maintain `u2u_host_utmi_linestate` and `u2u_dev_utmi_linestate` to SE0 irrespective of opcode from the device and host controllers.
2. The U2U Bridge drives `u2u_dev_utmi_linestate` to SE0 state (2'b00) if the device is not attached. Otherwise, it drives J state (2'b01) as indicated previously.
3. The host controller waits for `host_utmiotg_vbusvalid` =1.
4. The host controller combinatorially de-asserts `host_u2u_utmi_reset`, `host_u2u_utmi_suspend_n` and `host_u2u_utmi_l1_suspend_n` to the U2U Bridge allowing the U2U Bridge to assert `utmiclk_req_host` request.
5. SoC switches on `utmiclk_host`, and asserts `utmiclk_ack_host`.
6. The host controller is fed with `u2u_host_utmi_clk`. A clock gate is present within the U2U Bridge to block this clock in L1 and L2 suspend.
7. The host controller drives `host_u2u_utmi_xcvrselect` to 2'b01 (FS mode) and `host_u2u_utmi_opmode` to 2'b00, and waits for J (2'b01) on `u2u_host_utmi_linestate`.
8. The U2U Bridge drives `u2u_dev_utmi_linestate` and `u2u_host_utmi_linestate` to J state (2'b01) after it detects device attach (`dev_u2u_utmi_xcvrselect` = 2'b01 (FS mode) and `dev_u2u_utmi_opmode` = 2'b00) provided `soc_vbusvalid` is asserted.
9. When the host controller detects J (2'b01) on `u2u_host_utmi_linestate`, it asserts host HS reset by driving `host_u2u_utmi_xcvrselect` to 2'b00 (HS) and `host_u2u_utmi_opmode` to 2'b10 with `host_u2u_utmi_txvalid` = 0.

The U2U Bridge relies on `host_u2u_utmi_opmode` = 2'b10 with `xcvrselect` = 00 to infer reset. It ignores `host_u2u_utmi_txvalid`.

10. The U2U Bridge drives `u2u_dev_utmi_linestate` to SE0 state (2'b00).

The device detects the reset and does not respond with a Chirp. Therefore, the host will operate in FS mode.

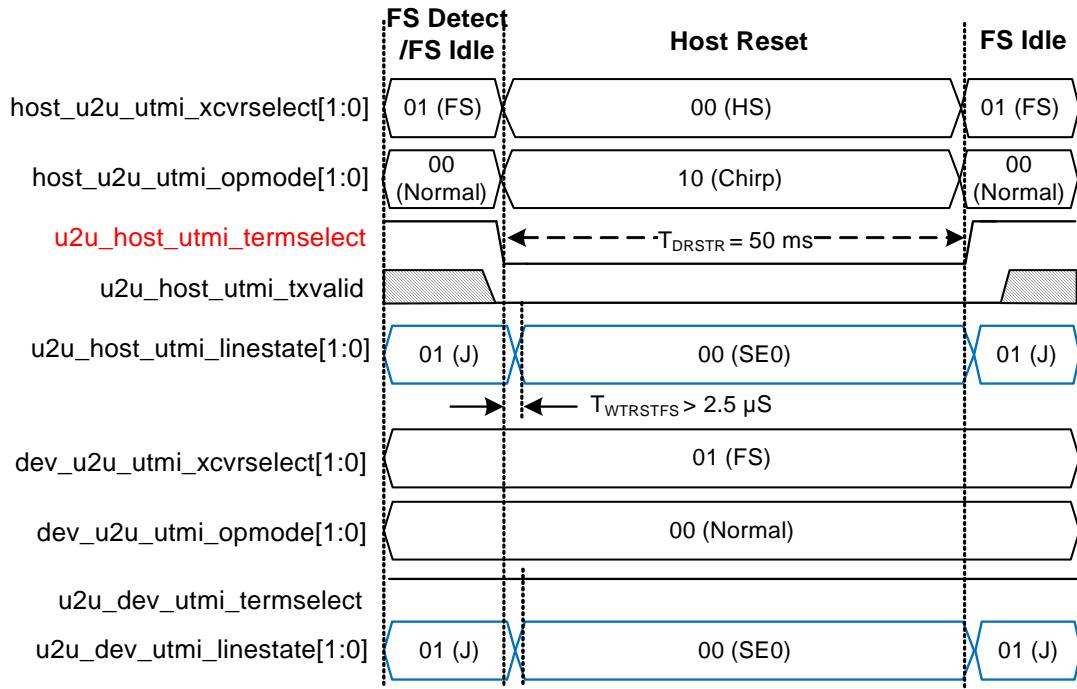


**Note** Under all scenarios, when `soc_vbusvalid` is de-asserted, the U2U Bridge drives `u2u_host_utmi_linestate` and `u2u_dev_utmi_linestate` to SE0 irrespective of opcode and `xcvrselect` from the device and host controllers.

## J.2.2 Host Reset Sequence Following Initial Device Connect

Figure J-2 shows the host reset sequence following initial device connect.

**Figure J-2 Host Reset Following Initial Device Connect**



The sequence of events at the U2U Bridge for Host Reset following initial device connect is as follows:

- Initially, both device and host controllers drive their xxx\_u2u\_utmi\_xcvrselect to 2'b01 and xxx\_u2u\_utmi\_opmode to 2'b00.
- u2u\_dev\_utmi\_linestate and u2u\_host\_utmi\_linestate both reflect J state.
- The host controller detects J (2'b01) on u2u\_host\_utmi\_linestate.
- The host controller asserts host reset by driving host\_u2u\_utmi\_xcvrselect to 2'b00 and host\_u2u\_utmi\_opmode to 2'b10 (and host\_u2u\_utmi\_termselect to 1'b0).
- The U2U Bridge samples the Host Controller Reset for TWTRSTFS (min 2.5μs, max 3000μs; Refer to the *USB2.0 Specification*) by using the combination of host\_u2u\_utmi\_xcvrselect = 2'b00 and host\_u2u\_utmi\_opmode = 2'b10, and then drives u2u\_dev\_utmi\_linestate and u2u\_host\_utmi\_linestate to SE0 state (2'b00).
- The device controller continues to drive dev\_u2u\_utmi\_xcvrselect to 2'b01 and dev\_u2u\_utmi\_opmode to 2'b00.
- At the end of Host Reset period ( $T_{DRSTR} \geq 50 \text{ ms}$ ), the host controller drives host\_u2u\_utmi\_xcvrselect to 2'b01 and host\_u2u\_utmi\_opmode to 2'b00 (and host\_u2u\_utmi\_termselect to 1'b0) to drive FS idle (J) for a minimum of 10 ms reset recovery time ( $T_{RSTRCY}$ ) before it starts any traffic on the USB bus.
- The U2U Bridge drives u2u\_dev\_utmi\_linestate and u2u\_host\_utmi\_linestate to J state (2'b01).



- host\_u2u\_utmi\_txvalid is de-asserted throughout the Host Reset Sequence.
- A device operating in low-speed or full-speed mode that sees an SE0 on its upstream facing port for more than  $T_{DETRST} = 2.5\mu s$  must treat that signal as a reset.
- host\_u2u\_utmi\_termselect is not an input to the U2U Bridge. It is shown here for reference only.

### J.2.3 Host Reset Sequence During Normal Traffic

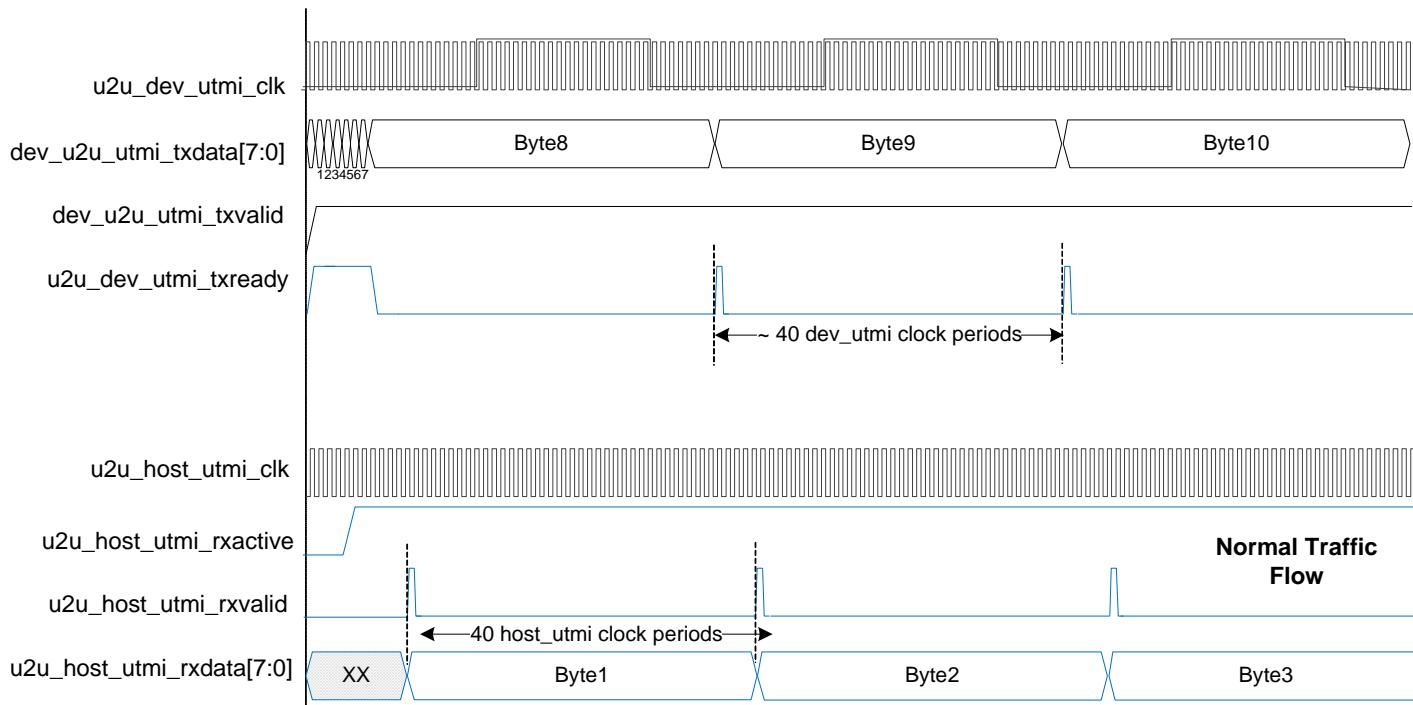
The sequence of events at the U2U Bridge for Host Reset during normal traffic is similar to the Host Reset following the initial device connect except that Step 5 is replaced as follows:

The U2U Bridge samples the Host Controller Reset for  $T_{FILTSE0}$  (min 2.5 $\mu s$ ; Refer to the *USB 2.0 Specification*), and then drives u2u\_dev\_utmi\_linestate and u2u\_host\_utmi\_linestate to SE0 state (2'b00).

### J.2.4 Normal Traffic Flow in FS Idle Following Host Reset

This section illustrates the flow of normal traffic from device to host. Similar flow applies for host to device traffic as well. [Figure J-3](#) shows the normal traffic flow in full-speed mode.

**Figure J-3** Normal Traffic Flow in FS Mode



Note that in FS mode, with utmi\_clk at 60MHz, there are 5 CLK cycles per FS bit time, typically 40 CLK cycles per FS byte time.

The traffic flow from device to host is as follows:

1. The U2U Bridge waits for dev\_u2u\_utmi\_txvalid to be asserted.
2. The U2U Bridge has an elasticity buffer which is a clock domain crossing FIFO. As long as the elasticity buffer is not full on the write-side of the buffer, the U2U Bridge asserts u2u\_dev\_utmi\_txready if dev\_u2u\_utmi\_txvalid is asserted and writes the data present on dev\_u2u\_utmi\_txdata into the elasticity buffer.
3. When the elasticity buffer is full, the U2U Bridge de-asserts u2u\_dev\_utmi\_txready until the elastic buffer is not full or when dev\_u2u\_utmi\_txvalid is asserted.
4. The read-side logic asserts rxactive as long as the FIFO is not empty on the read side.
5. The read-side logic waits until the number of bytes in the FIFO available on the read side is depth/2. When the read side has number of bytes in the FIFO greater than or equal to depth/2, it reads one byte from the FIFO.

In any case, if the read-side FIFO is not empty and there is no read for 40 utmiclk\_host cycles, the read-side logic will read one byte from the FIFO.

6. Each time the read-side logic reads from the FIFO, it asserts u2u\_host\_utmi\_rxvalid and presents the data read from the FIFO on u2u\_host\_utmi\_rxdata.

The following are the rules for normal traffic as per the *UTMI Specification*:

- rxactive, once asserted is not de-asserted until EOP.
- After EOP, rxactive is negated no more than 2 destination utmi\_clks after an FS Idle state is detected on the USB.
- rxactive is negated for at least 4 destination utmi\_clks between consecutive received packets.

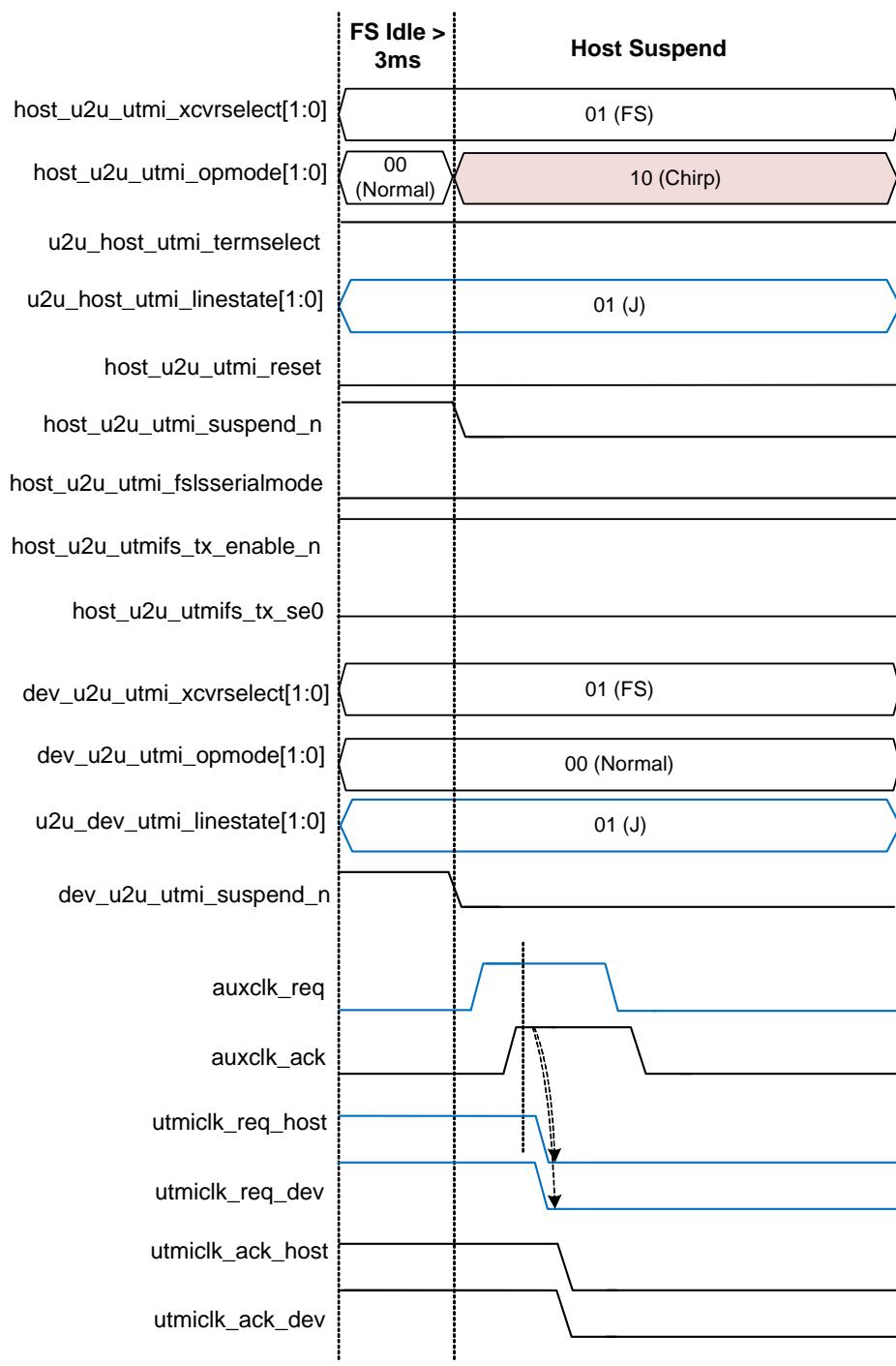
## J.2.5 Suspend Sequence

[Figure J-4](#) illustrates the suspend sequence.

Suspend sequence is as follows:

1. The host controller stops sending all packets including SOF to the device.
2. When the device controller sees inactivity on the bus in terms of FS idle for more than 3.0 ms, it enters suspend state and asserts dev\_u2u\_utmi\_suspend\_n.
3. The host controller simultaneously monitors the time of inactivity on the bus. When this exceeds 3.0 ms, it asserts host\_u2u\_utmi\_suspend\_n.
4. The U2U Bridge asserts auxclk\_req when either host\_u2u\_utmi\_suspend\_n or dev\_u2u\_utmi\_suspend\_n is asserted and waits for auxclk\_ack. When auxclk\_ack is asserted, it de-asserts utmiclk\_req\_host if host\_u2u\_utmi\_suspend\_n is asserted and similarly de-asserts utmiclk\_req\_dev if dev\_u2u\_utmi\_suspend\_n is asserted.
5. All internal logic within the U2U Bridge which is essential for operation during suspend operates in auxclk domain when suspend state is active.
6. The U2U Bridge stops u2u\_dev\_utmi\_clk and u2u\_host\_utmi\_clk clocks during suspended condition.
7. The assertion of either host\_u2u\_l1\_suspend\_n or host\_u2u\_suspend\_n from the host controller during suspend indicates that the parallel mode control signals host\_u2u\_xcvrselect[1:0], host\_u2u\_utmi\_opmode[1:0], and host\_u2u\_utmi\_termselect are not valid. The U2U Bridge monitors for changes in host\_u2u\_flslserialmode, host\_u2u\_utmifs\_tx\_enable\_n, and host\_u2u\_utmifs\_tx\_se0 for host exit from the suspend state. While host\_u2u\_utmi\_suspend\_n is asserted, the U2U Bridge ignores changes in host\_u2u\_xcvrselect[1:0] and host\_u2u\_utmi\_opmode[1:0], specifically, host\_u2u\_utmi\_opmode[1:0] driven to 10 (Chirp). This is an expected host controller behavior (to prepare (if) the interface needs to drive CHIRP K/J) and is tolerated by the U2U Bridge.
8. For device exit from the suspend state, the U2U Bridge continues to monitor the device-side parallel control interface for de-assertion of dev\_u2u\_utmi\_suspend\_n and change in dev\_u2u\_utmi\_opmode and dev\_u2u\_utmi\_txvalid.

The sequence of events shown in all suspend, resume, and remote wakeup scenarios are for L2 (normal suspend). For L1 (LPM), the sequence of events is similar, except that instead of waiting for 3 ms FS idle, there is an explicit exchange of LPM packets by the host and device controllers. When LPM packets are successfully exchanged, they enter L1 Suspend. During L1 Suspend, instead of indicating Suspend entry/exit through dev\_u2u\_utmi\_suspend\_n/host\_u2u\_utmi\_suspend\_n, the device/host controller will indicate L1 Suspend entry/exit through dev\_u2u\_utmi\_l1\_suspend\_n/ host\_u2u\_utmi\_l1\_suspend\_n respectively.

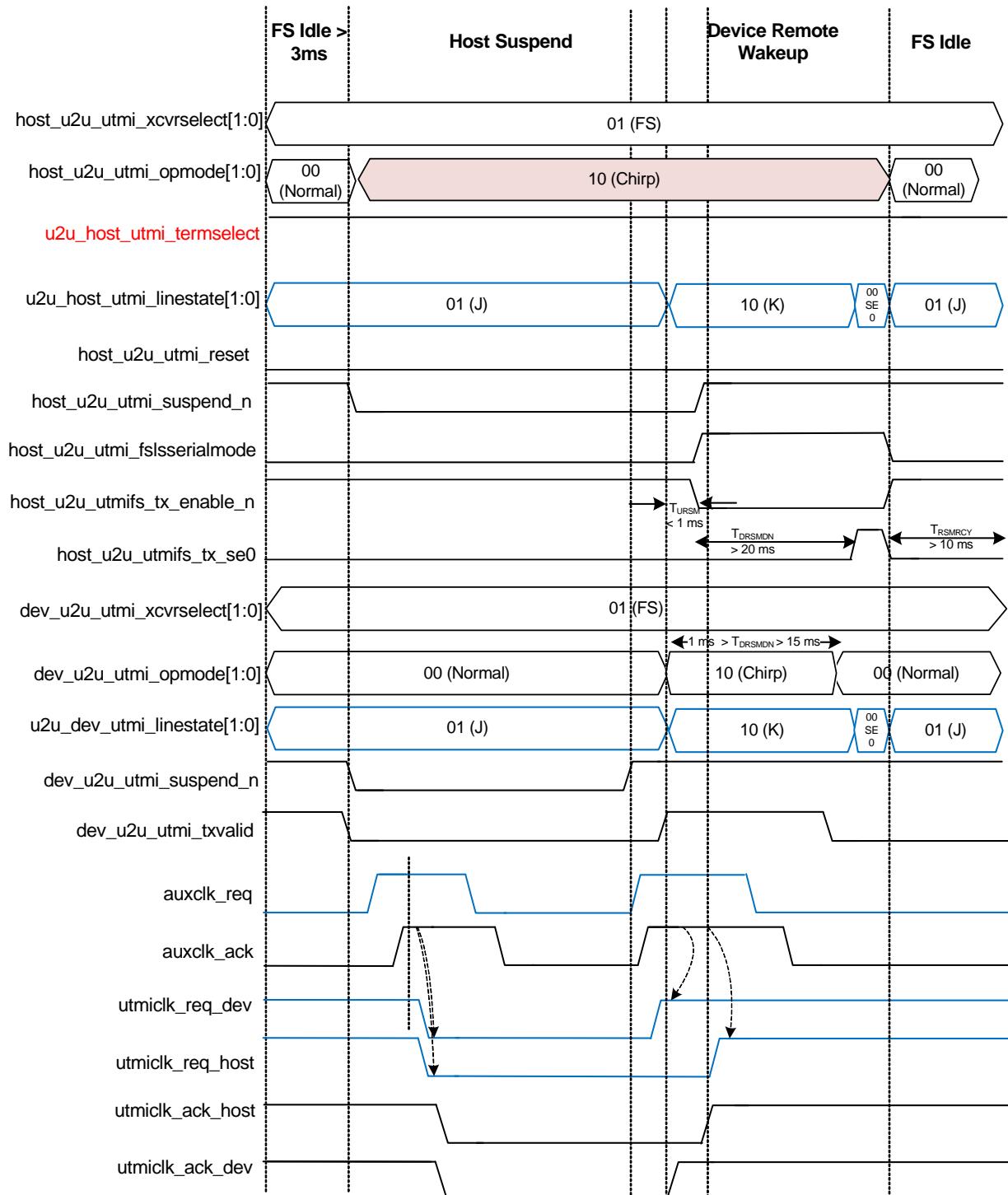
**Figure J-4 Suspend Sequence**

This diagram is only for illustration  
and not cycle accurate

## J.2.6 Device-Initiated Remote Wakeup Sequence

Figure J-5 illustrates device-initiated remote wakeup sequence.

**Figure J-5 Device-Initiated Remote Wakeup Sequence**



This diagram is only for illustration and is not cycle accurate.

A device with remote wakeup capability waits for at least 2 ms in suspend (total Idle state for  $T_{WTRMS} = 5$  ms) before it wakes up the host controller.

1. In suspend state, the device controller drives its control signals as follows:  
 $\text{dev\_u2u\_utmi\_xcvrselect} = 2'b01$  (FS);  $\text{dev\_u2u\_utmi\_opmode} = 2'b00$  (Normal);  
 $\text{dev\_u2u\_utmi\_termselect} = 1'b1$  (FS);  $\text{dev\_u2u\_utmi\_suspend\_n} = 1'b0$  (asserted);
2. For initiating remote wakeup, the device controller drives the following signals to the values listed below for 1 ms >  $T_{DRSMUP} > 15$  ms:  
 $\text{dev\_u2u\_utmi\_xcvrselect} = 2'b01$  (FS);  $\text{dev\_u2u\_utmi\_opmode} = 2'b10$  (Chirp);  
 $\text{dev\_u2u\_utmi\_termselect} = 1'b1$  (FS);  $\text{dev\_u2u\_utmi\_suspend\_n} = 1'b1$  (de-asserted);  
The device may drive the following signals:  
 $\text{dev\_u2u\_utmi\_txvalid} = 1'b1$  (Drive);  $\text{dev\_u2u\_utmi\_txdata[7:0]} = 8'h00$  (Drive);  
  - The U2U Bridge asserts  $\text{u2u\_dev\_utmi\_clk\_req}$  and  $\text{u2u\_host\_utmi\_clk\_req}$  to the SoC.
  - The U2U Bridge drives  $\text{u2u\_dev\_utmi\_linestate}$  and  $\text{u2u\_host\_utmi\_linestate}$  to 10 (K) in response and asserts  $\text{utmiclk\_req\_dev}$  and  $\text{utmiclk\_req\_host}$ .
3. At the end of this period, the device controller stops driving the bus (puts its drivers into the high-impedance state and does not drive the bus to the J state).  
 $\text{dev\_u2u\_utmi\_xcvrselect} = 2'b01$  (FS);  $\text{dev\_u2u\_utmi\_opmode} = 2'b00$  (Normal);  
 $\text{dev\_u2u\_utmi\_termselect} = 1'b1$  (FS);  $\text{dev\_u2u\_utmi\_suspend\_n} = 1'b1$  (de-asserted);  
 $\text{dev\_u2u\_utmi\_txvalid} = 1'b0$  (HighZ);
4. The host controller begins rebroadcast of resume signaling within a time  $T_{URSM} < 1$  ms after it detects  $\text{u2u\_host\_utmi\_linestate}$  as 10 (K) by driving its control signals as follows:
  - a.  $\text{host\_u2u\_utmi\_xcvrselect} = 2'b01$  (FS);  $\text{host\_u2u\_utmi\_opmode} = 2'b10$  (Chirp);  
 $\text{host\_u2u\_utmi\_termselect} = 1'b1$  (FS);  $\text{host\_u2u\_utmi\_suspend\_n} = 1'b0$  (de-asserted);  
 $\text{host\_u2u\_utmi\_fslsserialmode} = 1'b1$  (Serial);  $\text{host\_u2u\_utmifs\_tx\_enable\_n} = 1'b0$ ;  
 $\text{host\_u2u\_utmifs\_tx\_se0} = 1'b0$ ; For a period  $T_{DRSMDN} > 20$  ms
  - b.  $\text{host\_u2u\_utmi\_xcvrselect} = 2'b01$  (FS);  $\text{host\_u2u\_utmi\_opmode} = 2'b10$  (Chirp);  
 $\text{host\_u2u\_utmi\_termselect} = 1'b1$  (FS);  $\text{host\_u2u\_utmi\_suspend\_n} = 1'b0$  (de-asserted);  
 $\text{host\_u2u\_utmi\_fslsserialmode} = 1'b1$  (Serial);  $\text{host\_u2u\_utmifs\_tx\_enable\_n} = 1'b0$ ;  
 $\text{host\_u2u\_utmifs\_tx\_se0} = 1'b1$ ; For a period = 2 LS clock periods
5. The host controller stops resume signaling.

$\text{host\_u2u\_utmi\_xcvrselect} = 2'b01$  (FS);  $\text{host\_u2u\_utmi\_opmode} = 2'b00$  (Normal);  
 $\text{host\_u2u\_utmi\_termselect} = 1'b1$  (FS);  $\text{host\_u2u\_utmi\_suspend\_n} = 1'b0$  (de-asserted);  
 $\text{host\_u2u\_utmi\_fslsserialmode} = 1'b0$  (Parallel);  $\text{host\_u2u\_utmifs\_tx\_enable\_n} = 1'b1$ ;  
 $\text{host\_u2u\_utmifs\_tx\_se0} = 1'b0$ ; For a period  $T_{RSTRCY} > 10$  ms reset recovery time before the host controller starts any traffic on the USB bus.

Before the start of the resume signaling, when the device controller de-asserts  $\text{dev\_u2u\_utmi\_suspend\_n}$  (or  $\text{dev\_u2u\_utmi\_l1\_suspend\_n}$ ), the U2U Bridge will assert  $\text{auxclk\_req}$  and waits for  $\text{auxclk\_ack}$ .

The bridge then asserts  $\text{utmiclk\_req\_dev}$  and waits for the  $\text{utmi\_clk\_ack\_dev}$ . Once the device side of the bridge has a stable  $\text{utmi\_clk}$  input, the bridge will de-assert the  $\text{auxclk\_req}$  if no other overlapping

conditions for asserting auxclk\_req are present such as the de-assertion of host\_u2u\_utmi\_suspend\_n (or host\_u2u\_utmi\_l1\_suspend\_n).

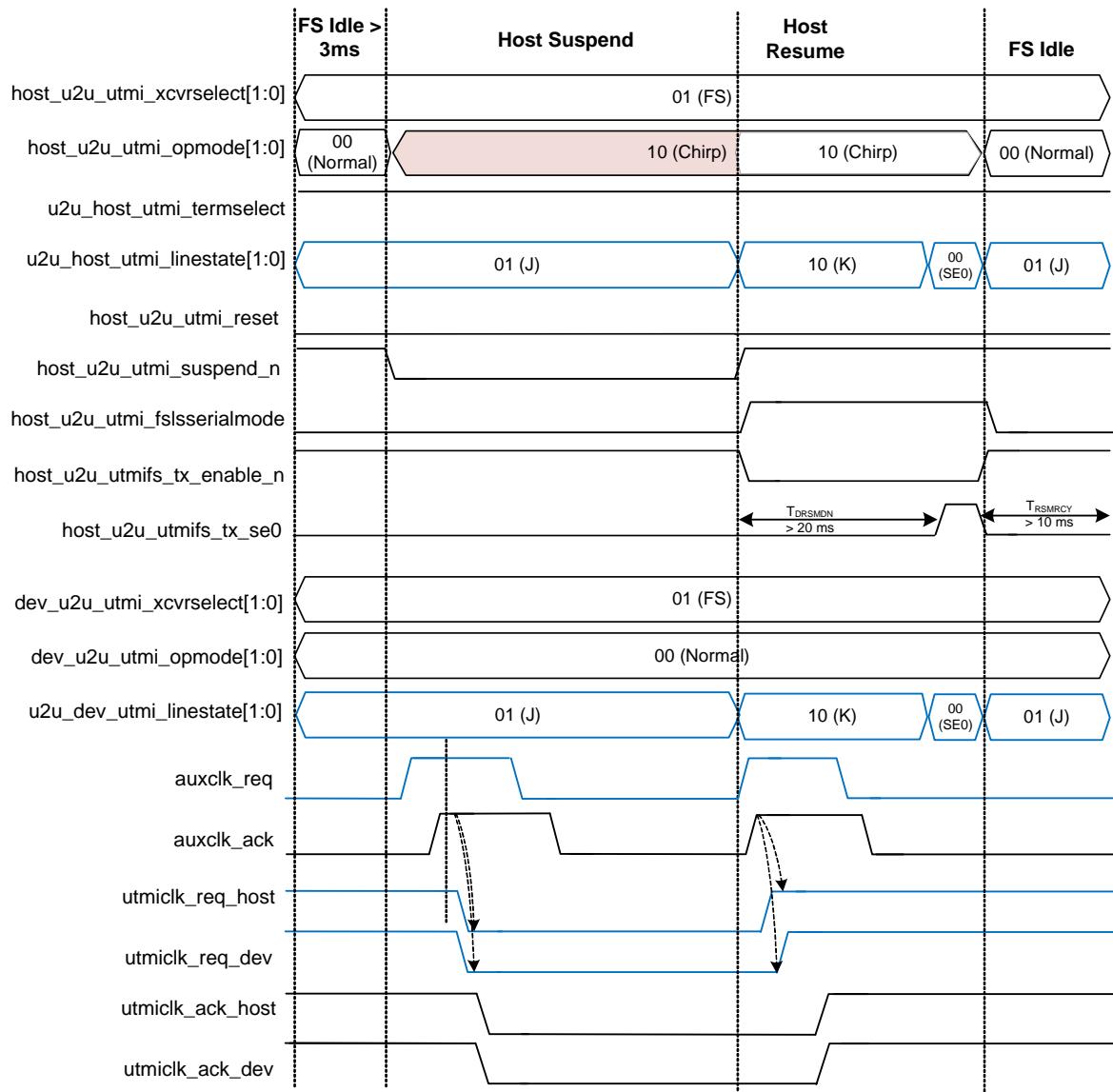


**Note** dev\_u2u\_utmi\_termselect is not an input to the U2U Bridge. In this section, it is used only for providing clarification.

## J.2.7 Host-Initiated Resume Sequence

Figure J-6 illustrates host-initiated resume sequence.

**Figure J-6 Host-Initiated Resume Sequence**



This diagram is only for illustration and is not cycle accurate.

1. In suspend state, the host controller drives its control signals as follows:

```
host_u2u_utmi_xcvrselect = 2'b01 (FS); host_u2u_utmi_opmode = 2'b10 (Chirp);
host_u2u_utmi_termselect = 1'b1 (FS); host_u2u_utmi_suspend_n = 1'b0 (asserted);
host_u2u_utmi_fslserialmode = 1'b0 (Parallel); host_u2u_utmifs_tx_enable_n = 1'b1;
host_u2u_utmifs_tx_se0 = 1'b0;
```

The U2U Bridge monitors for changes in host\_u2u\_fslserialmode, host\_u2u\_utmifs\_tx\_enable\_n and host\_u2u\_utmifs\_tx\_se0 for host exit from the suspend state. While host\_u2u\_utmi\_suspend\_n is asserted, the U2U Bridge ignores changes in host\_u2u\_xcvrselect[1:0] and host\_u2u\_utmi\_opmode[1:0], specifically, host\_u2u\_utmi\_opmode[1:0] driven to 10 (Chirp). This is an expected host controller behavior (to prepare (if) the interface needs to drive CHIRP K/J) and is tolerated by the U2U Bridge.

2. For initiating resume, the host controller drives the level of the following signals as listed below for  $T_{DRSMDN}$ :

```
host_u2u_utmi_fslserialmode = 1'b1 (Serial); host_u2u_utmifs_tx_enable_n = 1'b0;
host_u2u_utmifs_tx_se0 = 1'b0; host_u2u_utmi_suspend_n = 1'b1 (de-asserted)
```

The host may either update its opmode to 2'b10 (K) or continue to drive it as 2'b00 (Normal). Therefore, the U2U Bridge ignores opmode when host\_u2u\_utmi\_fslserialmode = 1'b1.

In response, the U2U Bridge drives u2u\_dev\_utmi\_linestate and u2u\_host\_utmi\_linestate to 10 (K) and asserts utmiclk\_req\_dev and utmiclk\_req\_host.

3. The host controller drives host\_u2u\_utmifs\_tx\_se0 to 1'b1 for 2 LS clock cycles period before driving it to 1'b0.

In response, the U2U Bridge drives u2u\_dev\_utmi\_linestate and u2u\_host\_utmi\_linestate to 00 (SE0).

4. The host controller then drives its control signals as follows to signal FS Idle:

```
host_u2u_utmi_xcvrselect = 2'b01 (FS); host_u2u_utmi_opmode = 2'b00 (Normal);
host_u2u_utmi_termselect = 1'b1 (FS); host_u2u_utmi_suspend_n = 1'b1 (de-asserted);
host_u2u_utmi_fslserialmode = 1'b0 (Parallel); host_u2u_utmifs_tx_enable_n = 1'b1;
host_u2u_utmifs_tx_se0 = 1'b0;
```

5. The previously-mentioned condition is maintained for a minimum of 10 ms resume recovery time ( $T_{RSMRCY}$ ) before the host controller starts any traffic on the USB bus.

Before the start of the resume signaling, when the host controller de-asserts host\_u2u\_utmi\_suspend\_n (or host\_u2u\_utmi\_l1\_suspend\_n), the U2U Bridge will assert auxclk\_req and waits for auxclk\_ack.

The bridge then asserts utmiclk\_req\_host and waits for the utmi\_clk\_ack\_host. Once the device side of the bridge has a stable utmi\_clk input, the bridge will de-assert the auxclk\_req if no other overlapping conditions for asserting auxclk\_req are present such as the de-assertion of dev\_u2u\_utmi\_suspend\_n (or dev\_u2u\_utmi\_l1\_suspend\_n).

## J.2.8 Host-Initiated Resume Sequence in Parallel Mode

The U2U Bridge provides support for host-resume operation through the parallel UTMI signals instead of using the serial mode signals.

In this case, the serial mode signals are inactive, that is, `host_u2u_utmi_fslsserialmode = 1'b0` (Serial), `host_u2u_utmifs_tx_enable_n = 1'b1`, and `host_u2u_utmifs_tx_se0 = 1'b0`.

When the U2U Bridge detects the host resume condition, it ensures that the host linestate is driven to the required values until the resume operation is completed.

1. In suspend state, the host controller drives its control signals as follows:

```
host_u2u_utmi_xcvrselect = 2'b01 (FS); host_u2u_utmi_opmode = 2'b10 (Chirp);
host_u2u_utmi_termselect = 1'b1 (FS); host_u2u_utmi_suspend_n = 1'b0 (asserted);
host_u2u_utmi_fslsserialmode = 1'b0 (Parallel); host_u2u_utmifs_tx_enable_n = 1'b1;
host_u2u_utmifs_tx_se0 = 1'b0;
```

The U2U Bridge monitors for changes in `host_u2u_fslsserialmode`, `host_u2u_utmifs_tx_enable_n`, and `host_u2u_utmifs_tx_se0` for host exit from the suspend state. While `host_u2u_utmi_suspend_n` is asserted, the U2U Bridge ignores changes in `host_u2u_xcvrselect[1:0]` and `host_u2u_utmi_opmode[1:0]`, specifically, `host_u2u_utmi_opmode[1:0]` driven to 10 (Chirp). This is an expected host controller behavior (to prepare (if) the interface needs to drive CHIRP K/J) and is tolerated by the U2U Bridge.

2. For initiating resume, the host controller drives the following signals:

```
host_u2u_utmi_fslsserialmode = 1'b0 (Serial); host_u2u_utmifs_tx_enable_n = 1'b1;
host_u2u_utmifs_tx_se0 = 1'b0; host_u2u_utmi_suspend_n = 1'b1 (de-asserted)
```

The host changes its opmode to 2'b10 (K). This is an indication to the U2U Bridge that the host-resume has begun. The U2U Bridge continues to hold the linestate at a value of 2'b01 until the `host_u2u_utmi_txvalid` signal is asserted. It asserts `utmiclk_req_dev` and `utmiclk_req_host`.

On assertion of the `host_u2u_utmi_txvalid` signal (= 1'b1), the U2U Bridge drives `u2u_dev_utmi_linestate` and `u2u_host_utmi_linestate` to 10 (K) in response.

3. Following this, the `host_u2u_utmi_txvalid` signal is de-asserted, but the host controller continues to hold the `host_u2u_utmi_opmode` signal at 2'b10.

In response, the U2U Bridge drives `u2u_dev_utmi_linestate` and `u2u_host_utmi_linestate` to 00 (SE0) for a period of 2\*LS BIT times. Then, it drives `u2u_dev_utmi_linestate` and `u2u_host_utmi_linestate` to 01 (J) for a period of 1\*LS BIT time.

4. The host controller then drives the following signals to signal FS Idle:

```
host_u2u_utmi_xcvrselect = 2'b01 (FS); host_u2u_utmi_opmode = 2'b00 (Normal);
host_u2u_utmi_termselect = 1'b1 (FS); host_u2u_utmi_suspend_n = 1'b1 (de-asserted);
host_u2u_utmi_fslsserialmode = 1'b0 (Parallel); host_u2u_utmifs_tx_enable_n = 1'b1;
host_u2u_utmifs_tx_se0 = 1'b0;
```

5. The previously-mentioned condition is maintained for a minimum of 10 ms resume recovery time ( $T_{RSMRCY}$ ) before the host controller starts any traffic on the USB bus.

Before the start of the resume signaling, when the host controller de-asserts `host_u2u_utmi_suspend_n` (or `host_u2u_utmi_l1_suspend_n`), the U2U Bridge will assert `auxclk_req` and waits for `auxclk_ack`.

The bridge then asserts utmiclk\_req\_host and the waits for the utmi\_clk\_ack\_host. Once the device side of the bridge has a stable utmi\_clk input, the bridge will de-assert the auxclk\_req if no other overlapping conditions for asserting auxclk\_req are present such as the de-assertion of dev\_u2u\_utmi\_suspend\_n (or dev\_u2u\_utmi\_l1\_suspend\_n)

## J.2.9 Host Reset During Suspend Sequence

Host Reset following a Suspend will occur through the parallel interface (see [Figure J-1](#)).

1. In suspend state, the host controller drives its control signals as follows:

```
host_u2u_utmi_xcvrselect = 2'b01 (FS); host_u2u_utmi_opmode = 2'b10 (Chirp);
host_u2u_utmi_termselect = 1'b1 (FS); host_u2u_utmi_suspend_n = 1'b0 (asserted);
host_u2u_utmi_fslsserialmode = 1'b0 (Parallel); host_u2u_utmifs_tx_enable_n = 1'b1;
host_u2u_utmifs_tx_se0 = 1'b0;
```

2. For initiating reset, it drives the level of the following signals as listed below for a minimum period of 50 ms ( $T_{DRSTR}$ ):

```
host_u2u_utmi_xcvrselect = 2'b00 (HS); host_u2u_utmi_opmode = 2'b10 (Chirp);
host_u2u_utmi_termselect = 1'b0 (HS); host_u2u_utmi_suspend_n = 1'b0 (de-asserted);
host_u2u_utmi_fslsserialmode = 1'b1 (Parallel); host_u2u_utmifs_tx_enable_n = 1'b1;
host_u2u_utmifs_tx_se0 = 1'b0; host_u2u_utmi_txvalid = 1'b0 (No drive)
```

The U2U Bridge drives u2u\_dev\_utmi\_linenstate and u2u\_host\_utmi\_linenstate to 00 (SE0) in response after seeing that host\_u2u\_utmi\_xcvrselect = 2'b00 (HS), host\_u2u\_utmi\_opmode = 2'b10 (Chirp), host\_u2u\_utmi\_suspend\_n = 1'b0 (de-asserted), and host\_u2u\_utmi\_fslsserialmode = 1'b1 (Parallel). It also asserts utmiclk\_req\_dev and utmiclk\_req\_host.

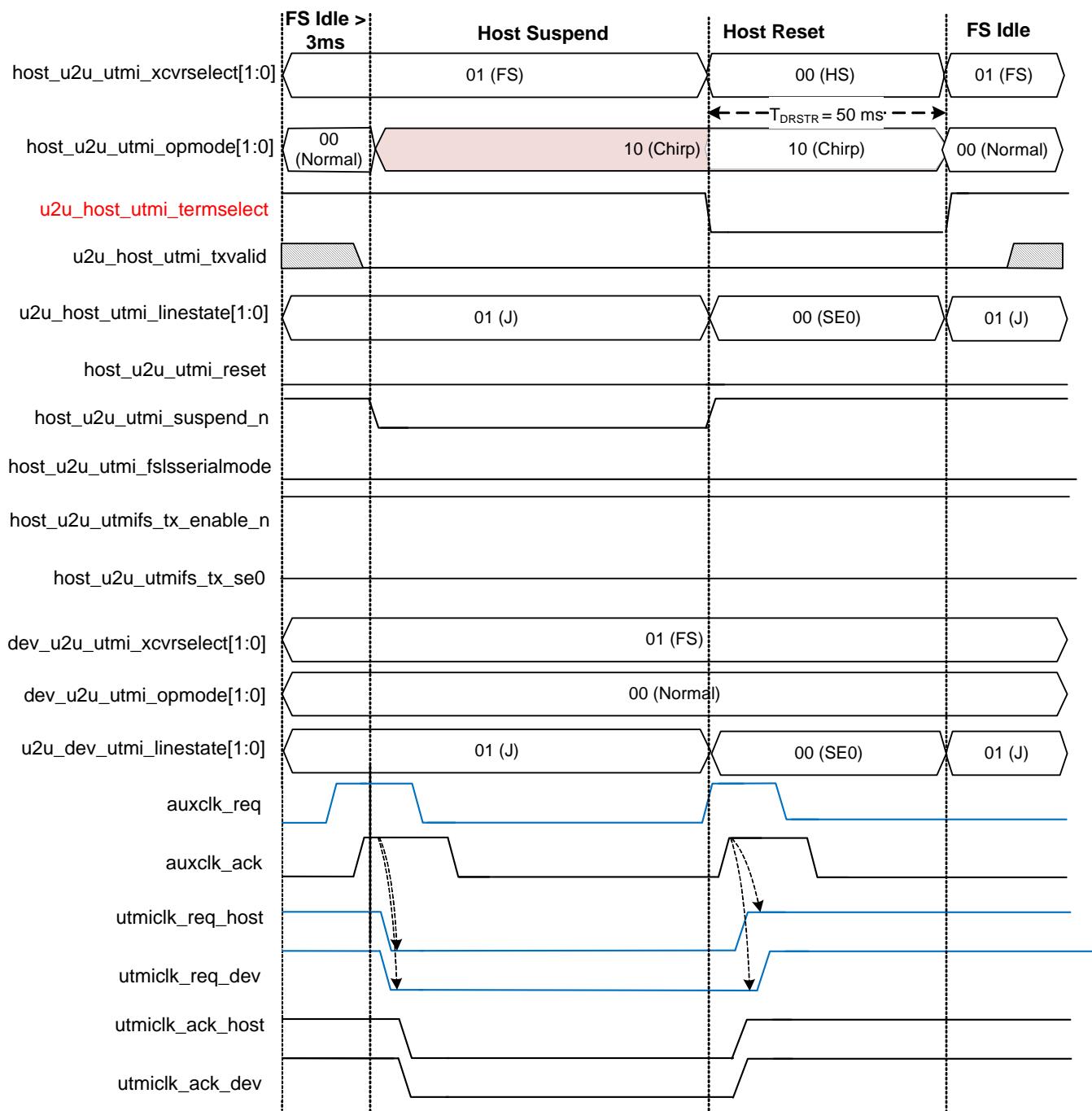
3. The host controller then drives its control signals as follows to signal FS Idle:

```
host_u2u_utmi_xcvrselect = 2'b01 (FS); host_u2u_utmi_opmode = 2'b00 (Normal);
host_u2u_utmi_termselect = 1'b1 (FS); host_u2u_utmi_suspend_n = 1'b1 (de-asserted);
host_u2u_utmi_fslsserialmode = 1'b0 (Parallel); host_u2u_utmifs_tx_enable_n = 1'b1;
host_u2u_utmifs_tx_se0 = 1'b0;
```

4. The above condition is maintained for a minimum of 10 ms reset recovery time ( $T_{RSTRCY}$ ) before the host controller starts any traffic on the USB bus.

Before the start of the USB reset signaling, when the host controller de-asserts host\_u2u\_utmi\_suspend\_n (or host\_u2u\_utmi\_l1\_suspend\_n), the U2U Bridge will assert auxclk\_req and waits for auxclk\_ack.

The bridge then asserts utmiclk\_req\_host and the waits for the utmi\_clk\_ack\_host. Once the device side of the bridge has a stable utmi\_clk input, the bridge will de-assert the auxclk\_req if no other overlapping conditions for asserting auxclk\_req are present such as the de-assertion of dev\_u2u\_utmi\_suspend\_n (or dev\_u2u\_utmi\_l1\_suspend\_n)

**Figure J-7 Host Reset During Suspend Through Parallel interface**

This diagram is only for illustration and is not cycle accurate.

## J.2.10 Disconnect Sequence

This section discusses the disconnect sequence.

### J.2.10.1 Host Disconnect

1. System indicates host disconnect by de-asserting soc\_vbusvalid.
2. When soc\_vbusvalid is de-asserted, the U2U Bridge sets u2u\_dev\_utmi\_linestate and u2u\_host\_utmi\_linestate to 2'b00. It also drives u2u\_dev\_vbusvalid = 0 and u2u\_dev\_iddig = 0.

### J.2.10.2 Device Disconnect

1. The device controller indicates device disconnect by driving dev\_u2u\_dev\_opmode[1:0] to NON\_DRV when soc\_vbusvalid is 1'b1.
2. The U2U Bridge sets the u2u\_dev\_utmi\_linestate and u2u\_host\_utmi\_linestate to 2'b00 indicating device soft disconnect.

## J.3 U2U Bridge Clock Logic

This section discusses the following topics:

- “Clock Request/Ack” on page [791](#)
- “[u2u\\_dev\\_utmi\\_clk](#)” on page [791](#)
- “[u2u\\_host\\_utmi\\_clk](#)” on page [791](#)
- “Internal Clocks” on page [792](#)

### J.3.1 Clock Request/Ack

The SoC must guarantee the following:

- auxclk is present when auxclk\_req is asserted and auxclk\_ack is asserted
- utmiclk\_host/utmiclk\_dev is present when corresponding utmiclk\_req is asserted and corresponding utmiclk\_ack is asserted

The SoC need not guarantee the following:

- auxclk is absent when auxclk\_req or auxclk\_ack is de-asserted
- utmiclk\_host/utmiclk\_dev is absent after corresponding utmiclk\_req is de-asserted

### J.3.2 [u2u\\_dev\\_utmi\\_clk](#)

[u2u\\_dev\\_utmi\\_clk](#) is gated in each of the following conditions:

- Assertion of PMC\_DEEP\_RST\_B
- Assertion of dev\_u2u\_l1\_suspend\_n
- Assertion of dev\_u2u\_suspend\_n

[u2u\\_dev\\_utmi\\_clk](#) is not gated when soc\_vbusvalid is 1'b0.

### J.3.3 [u2u\\_host\\_utmi\\_clk](#)

[u2u\\_host\\_utmi\\_clk](#) is gated in each of the following conditions:

- Assertion of PMC\_DEEP\_RST\_B
- Assertion of host\_u2u\_l1\_suspend\_n
- Assertion of host\_u2u\_suspend\_n

[u2u\\_host\\_utmi\\_clk](#) is not gated when soc\_vbusvalid is 1'b0.

### J.3.4 Internal Clocks

The U2U Bridge normally uses utmiclk\_dev/utmiclk\_host for portions of logic interfacing with device/host respectively. When corresponding utmiclk is switched off, this logic works on auxclk. Internal clock MUXing is implemented between utmiclk and auxclk for this purpose.

**Figure J-8 Clock - Controller State Matrix**

DWC_U2UB State	Signal name of Input Clock sources			Signal name of Internal controller clocks		Signal name of Output clocks to host and device	
	auxclk (24MHz)	utmiclk_host (60MHz)	utmiclk_dev (60MHz)	host_clk	dev_clk	u2u_host_utmi_clk	u2u_dev_utmi_clk
L0 State	OFF	ON	ON	utmiclk_host	utmiclk_dev	utmiclk_host	utmiclk_dev
L1 Suspend state	ON	OFF	OFF	auxclk	auxclk	OFF	OFF
L2 Suspend state	ON	OFF	OFF	auxclk	auxclk	OFF	OFF
PMC_DEEP_RST_B=1'b1 (Active)	OFF	OFF	OFF	OFF	OFF	OFF	OFF
dev_u2u_utmi_reset=1'b1 (Active)	-	-	-	-	-	-	auxclk
host_u2u_utmi_rst=1'b1 (Active)	-	-	-	-	-	auxclk	-
soc_vbusvalid=1'b0 (Inactive)	ON	OFF	OFF	auxclk	auxclk	auxclk	auxclk

**Notes**

- When dev\_u2u\_utmi\_reset is active, the output u2u\_dev\_utmi\_clk is muxed to auxclk.
- When host\_u2u\_utmi\_rst is active, the output u2u\_host\_utmi\_clk is muxed to auxclk.
- The SoC will guarantee that auxclk is present when auxclk\_req is asserted and auxclk\_ack is asserted.
- The SoC will not guarantee that auxclk is absent when auxclk\_req or auxclk\_ack is de-asserted.
- The SoC will guarantee that utmiclk\_host/utmiclk\_dev is present when corresponding utmiclk\_req is asserted and corresponding utmiclk\_ack is asserted.
- The SoC will not guarantee that utmiclk\_host/utmiclk\_dev is absent after corresponding utmiclk\_req is de-asserted.
- Inactive soc\_vbusvalid (=1'b0) signal will not gate the output clocks.
- Internal clocks will switch to respective utmiclks only when both the respective utmiclk\_reqs and utmiclk\_acks are active (1'b1).

## J.4 U2U Bridge Reset Logic

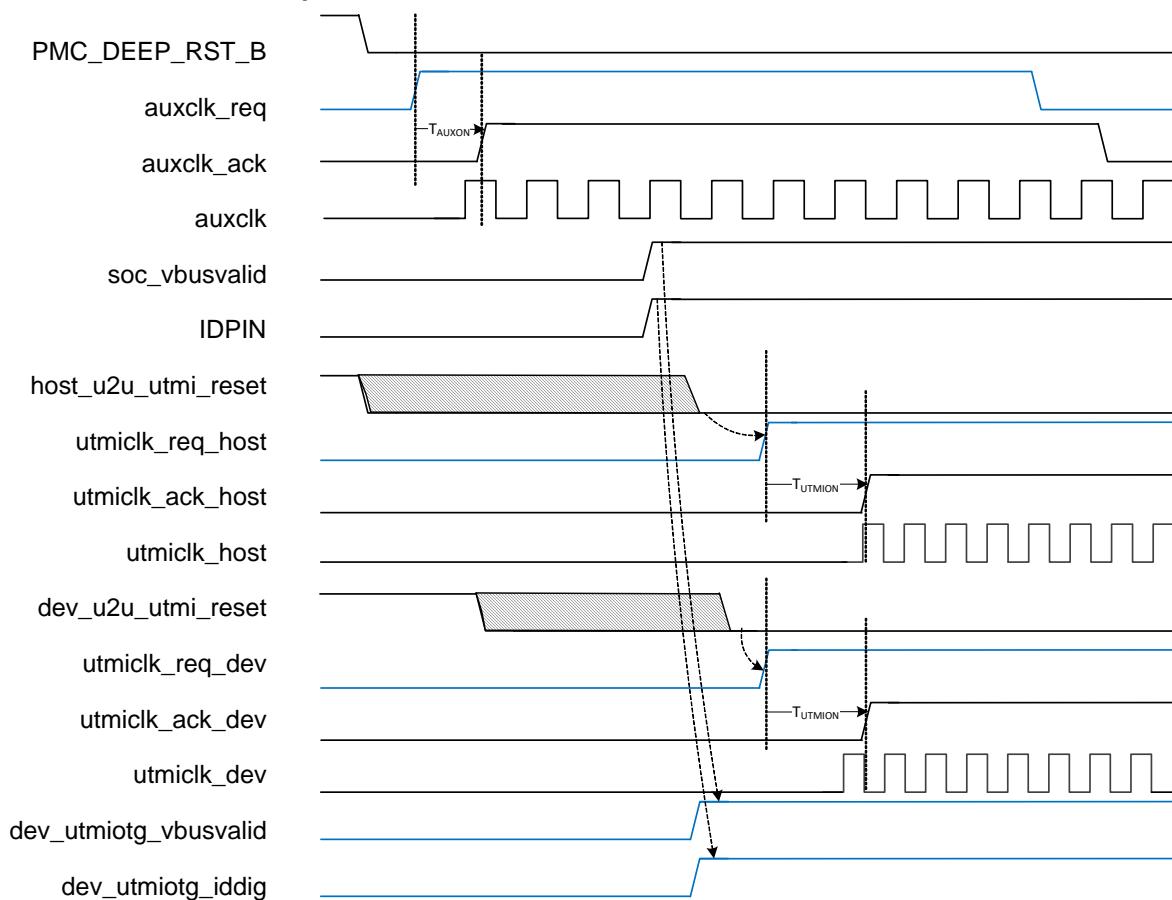
This section discusses the following topics:

- “Power-on Reset Sequence” on page [793](#)
- “PMC\_DEEP\_RST\_B” on page [794](#)
- “dev\_u2u\_utmi\_reset” on page [794](#)
- “host\_u2u\_utmi\_reset” on page [795](#)

### J.4.1 Power-on Reset Sequence

[Figure J-9](#) illustrates the power-on reset sequence.

**Figure J-9 Power-On Reset Sequence**



The power-on reset sequence is as follows:

1. Assert PMC\_DEEP\_RST\_B at power-on and then de-assert it.
2. The U2U Bridge ensures the corresponding power-on reset values for the following signals:
  - ❑ u2u\_host\_utmi\_linestate: SE0 state (2'b00)
  - ❑ u2u\_dev\_utmi\_linestate: SE0 state (2'b00)
  - ❑ u2u\_dev\_vbusvalid: 1'b0
  - ❑ u2u\_dev\_utmi\_txready, u2u\_host\_utmi\_txready: 1'b0
  - ❑ u2u\_dev\_utmi\_rxvalid, u2u\_host\_utmi\_rxvalid: 1'b0
  - ❑ u2u\_dev\_utmi\_rxactive, u2u\_host\_utmi\_rxactive: 1'b0
3. The U2U Bridge requests for auxclk by asserting auxclk\_req
4. The SoC responds by switching on the auxclk and asserts auxclk\_ack within time  $T_{AUXON}$ .
5. The U2U bridge waits for the condition soc\_vbusvalid = 1, to assert u2u\_dev\_vbusvalid/u2u\_dev\_iddig (These signals are generated in the auxclk domain).
6. The host/device controller de-asserts the UTMI resets (host\_u2u\_utmi\_reset / dev\_u2u\_utmi\_reset).
7. The U2U Bridge waits for host\_u2u\_utmi\_reset/dev\_u2u\_utmi\_reset de-assertion to request for corresponding UTMI clock (utmiclk\_req\_host/utmiclk\_req\_dev) provided the corresponding \*\_suspend\_n and \*l1\_suspend\_n signals are de-asserted.
8. The SoC responds by correspondingly switching on utmiclk\_host/utmiclk\_dev and asserting utmiclk\_ack\_host/utmiclk\_ack\_dev within time  $T_{UTMION}$ .
9. The U2U Bridge drives the UTMI clocks towards the corresponding host/device controllers (u2u\_host\_utmi\_clk/ u2u\_dev\_utmi\_clk).
10. The U2U Bridge de-asserts auxclk\_req.
11. The U2U Bridge starts to monitor host/device UTMI control signals to initiate the connect sequence.

#### J.4.2 **PMC\_DEEP\_RST\_B**

PMC\_DEEP\_RST\_B resets the complete U2U Bridge core.

#### J.4.3 **dev\_u2u\_utmi\_reset**

dev\_u2u\_utmi\_reset resets the following logic:

- FIFOs in the TX path of device UTMI interface
- FSM controlling the line\_state to the host and device UTMI interface
- Clock gating of u2u\_dev\_utmi\_clk (generation of utmiclk\_req\_dev)

dev\_u2u\_utmi\_reset does not affect auxclk gating (generation of auxclk\_req).

#### J.4.4 host\_u2u\_utmi\_reset

host\_u2u\_utmi\_reset resets the following logic:

- FIFOs in the TX path of host UTMI interface
- FSM controlling the line\_state to the host & device UTMI interface
- Clock gating of u2u\_host\_utmi\_clk (generation of utmiclk\_req\_host)

host\_u2u\_utmi\_reset does not reset the auxclk gating (generation of auxclk\_req).

---



**Note** The U2U Bridge does not expect host\_u2u\_utmi\_reset to be asserted while soc\_vbusvalid is asserted, except during suspend.

---

## J.5 Configuration Parameter

Table J-4 Parameters

Label	Description
U2UB Parallel Interface Data Bus Width	<p>Specifies the width of the Data bus.</p> <p><b>Values:</b> 8, 16</p> <p><b>Default Value:</b> 8</p> <p><b>Enabled:</b> Always</p> <p><b>Parameter Name:</b> DWC_U2UB_DATA_BUS_WIDTH</p>

## J.6 Signal Descriptions

This section details all possible I/O signals in the core. For configurable IP titles, your actual configuration might not contain all of these signals.

Inputs are on the left of the signal diagrams; outputs are on the right.

**Attention: For configurable IP titles, do not use this document to determine the exact I/O footprint of the core. It is for reference purposes only.**

When you configure the core in coreConsultant, you must access the I/O signals for your actual configuration at workspace/report/IO.html or workspace/report/IO.xml after you have completed the report creation activity. That report comes from the exact same source as this chapter but removes all the I/O signals that are not in your actual configuration. This does not apply to non-configurable IP titles. In addition, all parameter expressions are evaluated to actual values. Therefore, the widths might change depending on your actual configuration.

Some expressions might refer to TCL functions or procedures (sometimes identified as <functionof>) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the core in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

The I/O signals are grouped as follows:

- U2U Bridge: Device Controller Interface Descriptions on [page 798](#)
- U2U Bridge: Host Controller Interface Descriptions on [page 802](#)
- U2U Bridge: System Interface Descriptions on [page 806](#)
- U2U Bridge: Debug Port Interface Descriptions on [page 811](#)

### J.6.1 U2U Bridge: Device Controller Interface Descriptions Signals

dev_u2u_utmi_txvalidh -	- u2u_dev_utmi_clk
dev_u2u_utmi_l1_suspend_n -	- u2u_dev_utmi_linstate
dev_u2u_utmi_opmode -	- u2u_dev_utmi_rxactive
dev_u2u_utmi_suspend_n -	- u2u_dev_utmi_rxdata
dev_u2u_utmi_txdata -	- u2u_dev_utmi_rxvalid
dev_u2u_utmi_txvalid -	- u2u_dev_utmi_txready
dev_u2u_utmi_xcvrselect -	- u2u_dev_vbusvalid
	- u2u_dev_iddig
	- u2u_dev_utmi_rxvalididh

**Table J-5 U2U Bridge: Device Controller Interface Descriptions Signals**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
u2u_dev_utmi_clk	O	<p>Device Controller UTMI Clock. This clock is used by the protocol layer of the device controller. It is derived from the 60-MHz utmiclk_dev except in suspended state, when the output is derived from the 12-MHz auxclk.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> High <b>Synchronous to:</b> N/A <b>Registered:</b> N/A</p>
u2u_dev_utmi_linestate[1:0]	O	<p>Device Controller UTMI Line state. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_dev_utmi_clk <b>Registered:</b> No</p>
u2u_dev_utmi_rxactive	O	<p>Device Controller UTMI Receive Data Bus Active. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_dev_utmi_clk <b>Registered:</b> Yes</p>
u2u_dev_utmi_rxdata[(DWC_U2UB_DA TA_BUS_WIDTH-1):0]	O	<p>Device Controller UTMI Receive Data Bus. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_dev_utmi_clk <b>Registered:</b> Yes</p>
u2u_dev_utmi_rxvalid	O	<p>Device Controller UTMI Receive Data Bus Ready. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_dev_utmi_clk <b>Registered:</b> Yes</p>

**Table J-5 U2U Bridge: Device Controller Interface Descriptions Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
u2u_dev_utmi_txready	O	<p>Device Controller UTMI Transmit Ready. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_dev_utmi_clk <b>Registered:</b> Yes</p>
u2u_dev_vbusvalid	O	<p>Device Controller VBUS Valid. This signal indicates whether the VBUS is valid to the device controller.</p> <ul style="list-style-type: none"> <li>■ 1: VBUS is valid; Indicates that the host is active and the device can initiate a connection sequence with the host.</li> <li>■ 0: VBUS is not valid; Indicates that the host is inactive and the device must be in disconnected state.</li> </ul> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> High <b>Synchronous to:</b> Asynchronous <b>Registered:</b> No</p>
u2u_dev_iddig	O	<p>Device Controller ID pin. This signal indicates whether the connected plug is mini-A or mini-B.</p> <ul style="list-style-type: none"> <li>■ 1: Mini-B connector</li> <li>■ 0: Mini-A connector</li> </ul> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> N/A <b>Synchronous to:</b> Asynchronous <b>Registered:</b> No</p>
u2u_dev_utmi_rxvalidh	O	<p>Device Controller UTMI Receive Data Bus Ready for the high byte of the rxdata. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> (DWC_U2UB_DATA_BUS_WIDTH==16) <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_dev_utmi_clk <b>Registered:</b> Yes</p>

**Table J-5 U2U Bridge: Device Controller Interface Descriptions Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
dev_u2u_utmi_txvalidh	I	<p>Device Controller UTMI Transmit Data Bus Valid for the high byte of txdata.</p> <p>As per the <i>UTMI Specification</i>.</p> <p><i>Reset Value:</i> 1'b0</p> <p><b>Exists:</b> (DWC_U2UB_DATA_BUS_WIDTH==16)</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> As per the UTMI Specification</p> <p><b>Synchronous to:</b> u2u_dev_utmi_clk</p> <p><b>Registered:</b> Yes</p>
dev_u2u_utmi_l1_suspend_n	I	<p>Device Controller L1 Suspend.</p> <p>This signal indicates that the device controller is in L1 suspend state.</p> <ul style="list-style-type: none"> <li>■ 0: Device controller is in L1 suspend state</li> <li>■ 1: Device controller is not in L1 suspend state</li> </ul> <p><i>Reset Value:</i> 1'b1</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> Low</p> <p><b>Synchronous to:</b> Assertion synchronous to u2u_dev_utmi_clk. De-assertion asynchronous.</p> <p><b>Registered:</b> No</p>
dev_u2u_utmi_opmode[1:0]	I	<p>Device Controller UTMI Operational Mode.</p> <p>As per the <i>UTMI Specification</i>.</p> <p><i>Reset Value:</i> 2'b01</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> As per the UTMI Specification</p> <p><b>Synchronous to:</b> u2u_dev_utmi_clk</p> <p><b>Registered:</b> Yes</p>
dev_u2u_utmi_suspend_n	I	<p>Device Controller Suspend.</p> <p>This signal indicates that the device controller is in suspend state.</p> <ul style="list-style-type: none"> <li>■ 0: Device controller is in suspend state</li> <li>■ 1: Device controller is not in suspend state</li> </ul> <p><i>Reset Value:</i> 1'b1</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> Low</p> <p><b>Synchronous to:</b> Assertion synchronous to u2u_dev_utmi_clk. De-assertion asynchronous.</p> <p><b>Registered:</b> No</p>

**Table J-5 U2U Bridge: Device Controller Interface Descriptions Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
dev_u2u_utmi_txdata[(DWC_U2UB_DAT_A_BUS_WIDTH-1):0]	I	<p>Device Controller UTMI Transmit Data Bus. As per the <i>UTMI Specification</i>.</p> <p><i>Reset Value:</i> Don't care</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> As per the UTMI Specification</p> <p><b>Synchronous to:</b> u2u_dev_utmi_clk</p> <p><b>Registered:</b> Yes</p>
dev_u2u_utmi_txvalid	I	<p>Device Controller UTMI Transmit Data Bus Valid. As per the <i>UTMI Specification</i>.</p> <p><i>Reset Value:</i> 1'b0</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> As per the UTMI Specification</p> <p><b>Synchronous to:</b> u2u_dev_utmi_clk</p> <p><b>Registered:</b> Yes</p>
dev_u2u_utmi_xcvrselect[1:0]	I	<p>Device Controller UTMI Transceiver Selection. As per the <i>UTMI Specification</i>.</p> <p><i>Reset Value:</i> Don't care</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> As per the UTMI Specification</p> <p><b>Synchronous to:</b> u2u_dev_utmi_clk</p> <p><b>Registered:</b> Yes</p>

### J.6.2 U2U Bridge: Host Controller Interface Descriptions Signals

host_u2u_utmi_txvalidh -	- u2u_host_utmi_clk
host_u2u_utmi_fslsserialmode -	- u2u_host_utmi_linestate
host_u2u_utmi_l1_suspend_n -	- u2u_host_utmi_rxactive
host_u2u_utmi_opmode -	- u2u_host_utmi_rxdata
host_u2u_utmi_reset -	- u2u_host_utmi_rxvalid
host_u2u_utmi_suspend_n -	- u2u_host_utmi_txready
host_u2u_utmi_txdata -	- u2u_host_utmi_txvalidh
host_u2u_utmi_txvalid -	
host_u2u_utmi_xcvrselect -	
host_u2u_utmifs_tx_enable_n -	
host_u2u_utmifs_tx_se0 -	

**Table J-6 U2U Bridge: Host Controller Interface Descriptions Signals**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
u2u_host_utmi_clk	O	<p>Host Controller UTMI Clock. This clock is used by the protocol layer of the host controller. It is derived from the 60-MHz utmiclk_host except in the suspended state, when the output is derived from the 12-MHz auxclk.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> High <b>Synchronous to:</b> N/A <b>Registered:</b> N/A</p>
u2u_host_utmi_linestate[1:0]	O	<p>Host Controller UTMI Line state. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_host_utmi_clk <b>Registered:</b> No</p>
u2u_host_utmi_rxactive	O	<p>Host Controller UTMI Receive Data Bus Active. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_host_utmi_clk <b>Registered:</b> Yes</p>
u2u_host_utmi_rxdata[(DWC_U2UB_DA TA_BUS_WIDTH-1):0]	O	<p>Host Controller UTMI Receive Data Bus. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_host_utmi_clk <b>Registered:</b> Yes</p>
u2u_host_utmi_rxvalid	O	<p>Host Controller UTMI Receive Data Bus Ready. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_host_utmi_clk <b>Registered:</b> Yes</p>

**Table J-6 U2U Bridge: Host Controller Interface Descriptions Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
u2u_host_utmi_txready	O	<p>Host Controller UTMI Transmit Ready. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_host_utmi_clk <b>Registered:</b> Yes</p>
u2u_host_utmi_rxvalidh	O	<p>Host Controller UTMI Receive Data Bus Ready for the high byte of the rxdata. As per the <i>UTMI Specification</i>.</p> <p><b>Exists:</b> (DWC_U2UB_DATA_BUS_WIDTH==16) <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_host_utmi_clk <b>Registered:</b> Yes</p>
host_u2u_utmi_txvalidh	I	<p>Host Controller UTMI Transmit Data Bus Valid for the high byte of the txdata. As per the <i>UTMI Specification</i>. <i>Reset Value:</i> 1'b0</p> <p><b>Exists:</b> (DWC_U2UB_DATA_BUS_WIDTH==16) <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_host_utmi_clk <b>Registered:</b> Yes</p>
host_u2u_utmi_fslsserialmode	I	<p>Host Controller UTMI Full Speed Serial Mode Active. As per the <i>UTMI Specification</i>.</p> <p><i>Reset Value:</i> Don't care <b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> Asynchronous <b>Registered:</b> No</p>

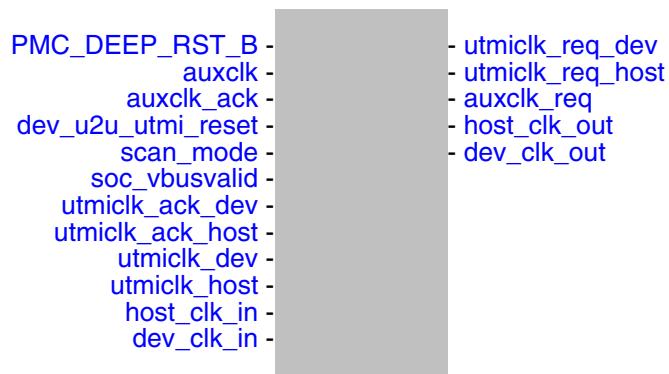
**Table J-6 U2U Bridge: Host Controller Interface Descriptions Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
host_u2u_utmi_l1_suspend_n	I	<p>Host Controller L1 Suspend. This signal indicates that the host controller is in L1 suspend state.</p> <ul style="list-style-type: none"> <li>■ 0: Host controller is in L1 suspend state</li> <li>■ 1: Host controller is not in L1 suspend state <i>Reset Value: 1'b1</i></li> </ul> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> Low <b>Synchronous to:</b> Assertion synchronous to u2u_host_utmi_clk. De-assertion asynchronous. <b>Registered:</b> No</p>
host_u2u_utmi_opmode[1:0]	I	<p>Host Controller UTMI Operational Mode. As per the <i>UTMI Specification</i>. <i>Reset Value: 2'b01</i></p> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> As per the UTMI Specification <b>Synchronous to:</b> u2u_host_utmi_clk <b>Registered:</b> Yes</p>
host_u2u_utmi_reset	I	<p>UTMI Host Controller Reset. This signal is the host UTMI PHY reset from the host controller.</p> <ul style="list-style-type: none"> <li>■ 1: Host UTMI PHY reset is asserted; The U2U Bridge host side logic TX data path and all FSMs will be reset.</li> <li>■ 0: Host UTMI PHY reset is de-asserted; The U2U Bridge host side logic will function normally. <i>Reset Value: 1'b1</i></li> </ul> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> High <b>Synchronous to:</b> Asynchronous <b>Registered:</b> No</p>
host_u2u_utmi_suspend_n	I	<p>Host Controller Suspend. This signal indicates that the host controller is in suspend state.</p> <ul style="list-style-type: none"> <li>■ 0: Host controller is in suspend state</li> <li>■ 1: Host controller is not in suspend state <i>Reset Value: 1'b1</i></li> </ul> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> Low <b>Synchronous to:</b> Assertion synchronous to u2u_host_utmi_clk. De-assertion asynchronous. <b>Registered:</b> No</p>

**Table J-6 U2U Bridge: Host Controller Interface Descriptions Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
host_u2u_utmi_txdata[(DWC_U2UB_DA TA_BUS_WIDTH-1):0]	I	<p>Host Controller UTMI Transmit Data Bus. As per the <i>UTMI Specification</i>.</p> <p><i>Reset Value:</i> Don't care</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> As per the UTMI Specification</p> <p><b>Synchronous to:</b> u2u_host_utmi_clk</p> <p><b>Registered:</b> Yes</p>
host_u2u_utmi_txvalid	I	<p>Host Controller UTMI Transmit Data Bus Valid. As per the <i>UTMI Specification</i>.</p> <p><i>Reset Value:</i> 1'b0</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> As per the UTMI Specification</p> <p><b>Synchronous to:</b> u2u_host_utmi_clk</p> <p><b>Registered:</b> Yes</p>
host_u2u_utmi_xcvrselect[1:0]	I	<p>Host Controller UTMI Transceiver Selection. As per the <i>UTMI Specification</i>.</p> <p><i>Reset Value:</i> Don't care</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> As per the UTMI Specification</p> <p><b>Synchronous to:</b> u2u_host_utmi_clk</p> <p><b>Registered:</b> Yes</p>
host_u2u_utmifs_tx_enable_n	I	<p>Host Controller UTMI Full Speed Transmit Enable. As per the <i>UTMI Specification</i>.</p> <p><i>Reset Value:</i> 1'b1</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> As per the UTMI Specification</p> <p><b>Synchronous to:</b> Asynchronous</p> <p><b>Registered:</b> No</p>
host_u2u_utmifs_tx_se0	I	<p>Host Controller UTMI Full Speed Transmit SE0. As per the <i>UTMI Specification</i>.</p> <p><i>Reset Value:</i> Don't care</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> As per the UTMI Specification</p> <p><b>Synchronous to:</b> Asynchronous</p> <p><b>Registered:</b> No</p>

### J.6.3 U2U Bridge: System Interface Descriptions Signals



**Table J-7 U2U Bridge: System Interface Descriptions Signals**

Port Name	I/O	Description
utmiclk_req_dev	O	<p>Device UTMI Clock Request. This signal is an indication from the U2U Bridge to turn ON the utmiclk_dev.</p> <ul style="list-style-type: none"> <li>■ 1: Device UTMI clock request is asserted.</li> <li>■ 0: Device UTMI clock request is de-asserted.</li> </ul> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> High <b>Synchronous to:</b> Asynchronous <b>Registered:</b> No</p>
utmiclk_req_host	O	<p>Host UTMI Clock Request. This signal is an indication from the U2U Bridge to turn ON the utmiclk_host.</p> <ul style="list-style-type: none"> <li>■ 1: Host UTMI clock request is asserted.</li> <li>■ 0: Host UTMI clock request is de-asserted.</li> </ul> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> High <b>Synchronous to:</b> Asynchronous <b>Registered:</b> No</p>

**Table J-7 U2U Bridge: System Interface Descriptions Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
auxclk_req	O	<p>Auxiliary Clock Request. This signal is an indication from the U2U Bridge to turn ON the auxclk.</p> <ul style="list-style-type: none"> <li>■ 1: Auxiliary clock request is asserted.</li> <li>■ 0: Auxiliary clock request is de-asserted.</li> </ul> <p><b>Exists:</b> Always  <b>Power Domain:</b> SINGLE_DOMAIN  <b>Active State:</b> High  <b>Synchronous to:</b> Asynchronous  <b>Registered:</b> No</p>
host_clk_out	O	<p>Host Pre-Scan Clock  <b>Exists:</b> (DWC_U2UB_EXT_SCAN_CLK==1)  <b>Power Domain:</b> SINGLE_DOMAIN  <b>Active State:</b> N/A  <b>Synchronous to:</b> N/A  <b>Registered:</b> N/A</p>
dev_clk_out	O	<p>Device Pre-Scan Clock  <b>Exists:</b> (DWC_U2UB_EXT_SCAN_CLK==1)  <b>Power Domain:</b> SINGLE_DOMAIN  <b>Active State:</b> N/A  <b>Synchronous to:</b> N/A  <b>Registered:</b> N/A</p>
PMC_DEEP_RST_B	I	<p>PMC Deep Reset. Power-on (deep) reset from the Power Management Controller of the SoC chip.</p> <ul style="list-style-type: none"> <li>■ 1: Power-on (deep) reset is asserted. The U2U Bridge will be completely reset.</li> <li>■ 0: Power-on (deep) reset is de-asserted. The U2U Bridge will function normally.</li> </ul> <p><b>Exists:</b> Always  <b>Power Domain:</b> SINGLE_DOMAIN  <b>Active State:</b> High  <b>Synchronous to:</b> Asynchronous  <b>Registered:</b> N/A</p>

**Table J-7 U2U Bridge: System Interface Descriptions Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
auxclk	I	<p>Auxiliary Clock.</p> <p>This clock is a 24-MHz +/- 2000 ppm auxiliary clock with 50% +/- 5% duty cycle. It is used for clock request interface, UTMI serial interface and other non-UTMI related logic.</p> <p><i>Note:</i> The utmiclk_dev, utmiclk_host, and auxclk are asynchronous to one another.</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> High</p> <p><b>Synchronous to:</b> N/A</p> <p><b>Registered:</b> N/A</p>
auxclk_ack	I	<p>Auxiliary Clock Request Acknowledge.</p> <p>This signal is an acknowledgment from the system that the auxclk is turned ON and it is valid within the required frequency/duty-cycle limits.</p> <ul style="list-style-type: none"> <li>■ 1: Auxiliary clock is valid.</li> <li>■ 0: Auxiliary clock is invalid. <i>Reset Value:</i> Don't care</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> High</p> <p><b>Synchronous to:</b> auxclk</p> <p><b>Registered:</b> No</p>
dev_u2u_utmi_reset	I	<p>UTMI Device Controller Reset.</p> <p>This is the device UTMI PHY reset from the system.</p> <ul style="list-style-type: none"> <li>■ 1: Device UTMI PHY reset is asserted; The U2U Bridge device side logic TX data path and all FSMs will be reset.</li> <li>■ 0: Device UTMI PHY Reset is de-asserted; The U2U Bridge device side logic will function normally. <i>Reset Value:</i> Don't care</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> High</p> <p><b>Synchronous to:</b> utmiclk_dev</p> <p><b>Registered:</b> N/A</p>

**Table J-7 U2U Bridge: System Interface Descriptions Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
scan_mode	I	<p>Scan Mode Port. This signal enables scan-mode bypass to improve scan coverage.</p> <ul style="list-style-type: none"> <li>■ 1'b0: Scan mode bypass disabled;</li> <li>■ 1'b1: Scan mode bypass enabled; Controls the internally generated clocks and reset during scan mode for clean DFT. During scan-mode, this signal should be set to 1.</li> </ul> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> High <b>Synchronous to:</b> N/A <b>Registered:</b> No</p>
soc_vbusvalid	I	<p>SoC VBUS Valid. This signal indicates whether the VBUS is valid.</p> <ul style="list-style-type: none"> <li>■ 1: VBUS is valid; Indicates that the host is active and device can initiate a connection sequence with host.</li> <li>■ 0: VBUS is not valid; Indicates that the host is inactive and device must be in disconnected state. <i>Reset Value:</i> 1'b0</li> </ul> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> High <b>Synchronous to:</b> Asynchronous <b>Registered:</b> Yes</p>
utmiclk_ack_dev	I	<p>Device UTMI Clock Request Acknowledge. This signal is an acknowledgment from the system that the utmiclk_dev is turned ON and it is valid within the frequency/duty-cycle limits defined in the <i>UTMI Specification</i>.</p> <ul style="list-style-type: none"> <li>■ 1: Device UTMI clock is valid.</li> <li>■ 0: Device UTMI clock is invalid. <i>Reset Value:</i> Don't care</li> </ul> <p><b>Exists:</b> Always <b>Power Domain:</b> SINGLE_DOMAIN <b>Active State:</b> High <b>Synchronous to:</b> utmiclk_dev <b>Registered:</b> No</p>

**Table J-7 U2U Bridge: System Interface Descriptions Signals (Continued)**

<b>Port Name</b>	<b>I/O</b>	<b>Description</b>
utmiclk_ack_host	I	<p>Host UTMI Clock Request Acknowledge.</p> <p>This signal is an acknowledgment from the system that the utmiclk_host is turned ON and it is valid within the frequency/duty-cycle limits defined in the <i>UTMI Specification</i>.</p> <ul style="list-style-type: none"> <li>■ 1: Host UTMI clock is valid.</li> <li>■ 0: Host UTMI clock is invalid. <i>Reset Value</i>: Don't care</li> </ul> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> High</p> <p><b>Synchronous to:</b> utmiclk_host</p> <p><b>Registered:</b> No</p>
utmiclk_dev	I	<p>Device UTMI Clock.</p> <p>This clock is a 60-MHz device UTMI clock with accuracy and duty cycle defined as per the <i>UTMI Specification</i>.</p> <p><i>Note:</i> The utmiclk_dev, utmiclk_host, and auxclk are asynchronous to one another.</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> High</p> <p><b>Synchronous to:</b> N/A</p> <p><b>Registered:</b> N/A</p>
utmiclk_host	I	<p>Host UTMI Clock.</p> <p>This clock is a 60-MHz host UTMI clock with accuracy and duty cycle defined as per the <i>UTMI Specification</i>.</p> <p><i>Note:</i> The utmiclk_dev, utmiclk_host, and auxclk are asynchronous to one another.</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> High</p> <p><b>Synchronous to:</b> N/A</p> <p><b>Registered:</b> N/A</p>
host_clk_in	I	<p>Host Post Scan Clock</p> <p><b>Exists:</b> (DWC_U2UB_EXT_SCAN_CLK==1)</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> N/A</p> <p><b>Synchronous to:</b> N/A</p> <p><b>Registered:</b> N/A</p>

**Table J-7 U2U Bridge: System Interface Descriptions Signals (Continued)**

Port Name	I/O	Description
dev_clk_in	I	<p>Device Post Scan Clock</p> <p><b>Exists:</b> (DWC_U2UB_EXT_SCAN_CLK==1)</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> N/A</p> <p><b>Synchronous to:</b> N/A</p> <p><b>Registered:</b> N/A</p>

#### J.6.4 U2U Bridge: Debug Port Interface Descriptions Signals

**Table J-8 U2U Bridge: Debug Port Interface Descriptions Signals**

Port Name	I/O	Description
u2u_soc_debug_out[(DWC_U2UB_DEB UG_OUT_BUS_WIDTH-1):0]	O	<p>U2U Bridge SoC Debug Out.</p> <p>For details, see 'Debug Port Details' section in the Databook.</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> N/A</p> <p><b>Synchronous to:</b> N/A</p> <p><b>Registered:</b> N/A</p>
soc_u2u_debug_in[(DWC_U2UB_DEBU G_INP_BUS_WIDTH-1):0]	I	<p>U2U Bridge SoC Debug In.</p> <p>For details, see 'Debug Port Details' section in the Databook.</p> <p><b>Exists:</b> Always</p> <p><b>Power Domain:</b> SINGLE_DOMAIN</p> <p><b>Active State:</b> N/A</p> <p><b>Synchronous to:</b> N/A</p> <p><b>Registered:</b> N/A</p>

#### Debug Port Details

Table J-9 on page 812 and Table J-10 on page 818 provide a detailed list of the debug input and outputs that are present in the u2u\_soc\_debug\_out and soc\_u2u\_debug\_in buses in the DWC\_U2UB controller.

**Table J-9 Description of Signals in u2u\_soc\_debug\_out**

<b>Bit Position</b>	<b>Output Port Name</b>	<b>Internal Signal Name</b>	<b>Clock Domain</b>	<b>Define Name for Bit Position and Width</b>	<b>Description</b>
0	dev_tx_fifo_RST_S_N	dev_RST_N	dev_clk	DWC_U2UB_DBGIO_DTX_FIFO_RST_S_N DWC_U2UB_DBGIO_DTX_FIFO_RST_S_N_W=1	Dev FIFO source domain reset signal
1	dev_tx_fifo_EMPTY_S	fifo_empty_S_unc conn	dev_clk	DWC_U2UB_DBGIO_DTX_FIFO_EMPTY_S DWC_U2UB_DBGIO_DTX_FIFO_EMPTY_S_W=1	Dev FIFO source domain FIFO empty signal
2	dev_tx_fifo_ALMOST_FULL_S	almost_full_S	dev_clk	DWC_U2UB_DBGIO_DTX_FIFO_AF_S DWC_U2UB_DBGIO_DTX_FIFO_AF_S_W=1	Dev FIFO source domain almost full signal
3	dev_tx_fifo_FULL_S	full_S_unconn	dev_clk	DWC_U2UB_DBGIO_DTX_FIFO_FULL_S DWC_U2UB_DBGIO_DTX_FIFO_FULL_S_W=1	Dev FIFO source domain full signal
4	dev_tx_fifo_CLR_S	dev_fifo_CLR_S	dev_clk	DWC_U2UB_DBGIO_DTX_FIFO_CLR_S DWC_U2UB_DBGIO_DTX_FIFO_CLR_S_W=1	Dev FIFO source domain FIFO clear signal
5	dev_tx_fifo_CLR_IN_PROG_D	clr_in_prog_d	host_clk	DWC_U2UB_DBGIO_DTX_FIFO_CLRINPROG_D DWC_U2UB_DBGIO_DTX_FIFO_CLRINPROG_D_W=1	Dev FIFO destination domain clear-in-progress signal
6	dev_tx_fifo_CLR_CMPLT_D	clr_cmplt_d	host_clk	DWC_U2UB_DBGIO_DTX_FIFO_CLR_CMPLT_D DWC_U2UB_DBGIO_DTX_FIFO_CLR_CMPLT_D_W=1	Dev FIFO destination domain clear completed signal
7	host_tx_fifo_RST_S_N	host_RST_N	host_clk	DWC_U2UB_DBGIO_HTX_FIFO_RST_D_N DWC_U2UB_DBGIO_HTX_FIFO_RST_D_N_W=1	Host FIFO source domain reset signal
8	host_tx_fifo_CLR_S	host_fifo_CLR_S	host_clk	DWC_U2UB_DBGIO_HTX_FIFO_CLR_S DWC_U2UB_DBGIO_HTX_FIFO_CLR_S_W=1	Host FIFO source domain FIFO-clear signal
9	host_tx_fifo_CLR_IN_PROG_D	clr_in_prog_d	dev_clk	DWC_U2UB_DBGIO_HTX_FIFO_CLRINPROG_D DWC_U2UB_DBGIO_HTX_FIFO_CLRINPROG_D_W=1	Host FIFO destination domain clear in progress signal

**Table J-9 Description of Signals in u2u\_soc\_debug\_out (Continued)**

<b>Bit Position</b>	<b>Output Port Name</b>	<b>Internal Signal Name</b>	<b>Clock Domain</b>	<b>Define Name for Bit Position and Width</b>	<b>Description</b>
10	host_tx_fifo_clr_cmplt_d	clr_cmplt_d	dev_clk	DWC_U2UB_DBGIO_HTX_FIFO_CLR_CMPLT_D DWC_U2UB_DBGIO_HTX_FIFO_CLR_CMPLT_D_W=1	Host FIFO destination domain clear completed signal
11	host_tx_fifo_empty_s	fifo_empty_s_unconn	host_clk	DWC_U2UB_DBGIO_HTX_FIFO_EMPTY_S DWC_U2UB_DBGIO_HTX_FIFO_EMPTY_S_W=1	Host FIFO source domain FIFO-empty signal
12	host_tx_fifo_almost_full_s	almost_full_s	host_clk	DWC_U2UB_DBGIO_HTX_FIFO_AF_S DWC_U2UB_DBGIO_HTX_FIFO_AF_S_W=1	Host FIFO source domain almost full signal
13	host_tx_fifo_full_s	full_s_unconn	host_clk	DWC_U2UB_DBGIO_HTX_FIFO_FULL_S DWC_U2UB_DBGIO_HTX_FIFO_FULL_S_W=1	Host FIFO source domain full signal
14	host_tx_fifo_clk_s	host_clk	host_clk	DWC_U2UB_DBGIO_HTX_FIFO_CLK_S DWC_U2UB_DBGIO_HTX_FIFO_CLK_S_W=1	Host FIFO source domain clk signal
15	host_tx_fifo_push_s	push_host_tx_data	host_clk	DWC_U2UB_DBGIO_HTX_FIFO_PUSH_S_N DWC_U2UB_DBGIO_HTX_FIFO_PUSH_S_N_W=1	Host FIFO source domain push data signal
24:16	host_tx_fifo_data_s[8:0]	data_s	host_clk	DWC_U2UB_DBGIO_HTX_FIFO_DATA_S DWC_U2UB_DBGIO_HTX_FIFO_DATA_S_W=9	Host FIFO source domain input data
27:25	host_tx_fsm_cs[2:0]	host_tx_fsm_cs	host_clk	DWC_U2UB_DBGIO_HTX_FSM_CS DWC_U2UB_DBGIO_HTX_FSM_CS_W=3	Host Tx pkt-emulator fsm_cs signal
29:28	host_tx_linestate_sync_eop[1:0]	host_tx_linestate_sync_eop	host_clk	DWC_U2UB_DBGIO_HTX_LS_SYNC_EOP DWC_U2UB_DBGIO_HTX_LS_SYNC_EOP_W=2	Host Tx pkt-emulator linestate output during sync-eop phase
30	mask_pop_d	mask_pop_d	dev_clk	DWC_U2UB_DBGIO_H_MASK_POP_D DWC_U2UB_DBGIO_H_MASK_POP_D_W=1	Signal to mask the pop signal in the device FIFO

**Table J-9 Description of Signals in u2u\_soc\_debug\_out (Continued)**

<b>Bit Position</b>	<b>Output Port Name</b>	<b>Internal Signal Name</b>	<b>Clock Domain</b>	<b>Define Name for Bit Position and Width</b>	<b>Description</b>
31	pop_data_in_wait	pop_data_in_wait	dev_clk	DWC_U2UB_DBGIO_H_POP_DATA_IN_WAIT DWC_U2UB_DBGIO_H_POP_DATA_IN_WAIT_W=1	Signal to enable pop from FIFO during the wait state in the device FIFO
32	host_tx_fifo_empty_d	empty_d	dev_clk	DWC_U2UB_DBGIOHTX_FIFO_EMPTY_D DWC_U2UB_DBGIOHTX_FIFO_EMPTY_D_W=1	Host FIFO destination domain empty signal
33	host_tx_fifo_almost_full_d	almost_full_d	dev_clk	DWC_U2UB_DBGIOHTX_FIFO_AF_D DWC_U2UB_DBGIOHTX_FIFO_AF_D_W=1	Host FIFO destination domain almost_full signal
34	dev_tx_fifo_clk_s	dev_clk	dev_clk	DWC_U2UB_DBGIODTX_FIFO_CLK_S DWC_U2UB_DBGIODTX_FIFO_CLK_S_W=1	Dev FIFO source domain clk signal
35	host_tx_fifo_pop_d	pop_d_masked	dev_clk	DWC_U2UB_DBGIOHTX_FIFO_POP_D_N DWC_U2UB_DBGIOHTX_FIFO_POP_D_N_W=1	Host FIFO destination domain pop signal
44:36	host_tx_fifo_data_d[8:0]	data_d	dev_clk	DWC_U2UB_DBGIOHTX_FIFO_DATA_D DWC_U2UB_DBGIOHTX_FIFO_DATA_D_W=9	Host FIFO destination domain pop data signal
47:45	dev_rx_fsm_cs[2:0]	dev_rx_fsm_cs	dev_clk	DWC_U2UB_DBGIODRX_FSM_CS DWC_U2UB_DBGIODRX_FSM_CS_W=3	Device RX pkt-emulator FSM-CS signal
49:48	dev_rx_linestate_sync_eop[1:0]	dev_rx_linestate_sync_eop	dev_clk	DWC_U2UB_DBGIODRX_LS_SYNC_EOP DWC_U2UB_DBGIODRX_LS_SYNC_EOP_W=2	Device RX pkt-emulator linestate signal during sync-eop phase
50	dev_rx_linestate_sync_eop_valid	dev_rx_linestate_sync_eop_valid	dev_clk	DWC_U2UB_DBGIODRX_LS_SYNC_EOP_VALID DWC_U2UB_DBGIODRX_LS_SYNC_EOP_VALID_W=1	Device RX pkt-emulator linestate valid signal

**Table J-9 Description of Signals in u2u\_soc\_debug\_out (Continued)**

<b>Bit Position</b>	<b>Output Port Name</b>	<b>Internal Signal Name</b>	<b>Clock Domain</b>	<b>Define Name for Bit Position and Width</b>	<b>Description</b>
51	dev_tx_fifo_push_s	push_dev_tx_data	dev_clk	DWC_U2UB_DBGIO_DTX_FIFO_PUSH_S_N DWC_U2UB_DBGIO_DTX_FIFO_PUSH_S_N_W=1	Dev FIFO source domain push signal
60:52	dev_tx_fifo_data_s[8:0]	data_s	dev_clk	DWC_U2UB_DBGIO_DTX_FIFO_DATA_S DWC_U2UB_DBGIO_DTX_FIFO_DATA_S_W=9	Dev FIFO source domain push data signal
61	dev_tx_fifo_empty_d	empty_d	host_clk	DWC_U2UB_DBGIO_DTX_FIFO_EMPTY_D DWC_U2UB_DBGIO_DTX_FIFO_EMPTY_D_W=1	Dev FIFO destination domain empty signal
62	dev_tx_fifo_almost_full_d	almost_full_d	host_clk	DWC_U2UB_DBGIO_DTX_FIFO_AF_D DWC_U2UB_DBGIO_DTX_FIFO_AF_D_W=1	Dev FIFO destination domain almost full signal
63	dev_tx_fifo_pop_d	pop_d_masked	host_clk	DWC_U2UB_DBGIO_DTX_FIFO_POP_D_N DWC_U2UB_DBGIO_DTX_FIFO_POP_D_N_W=1	Dev FIFO destination domain pop signal
72:64	dev_tx_fifo_data_d[8:0]	data_d	host_clk	DWC_U2UB_DBGIO_DTX_FIFO_DATA_D DWC_U2UB_DBGIO_DTX_FIFO_DATA_D_W=9	Dev FIFO destination domain pop data signal
73	mask_pop_d	mask_pop_d	host_clk	DWC_U2UB_DBGIO_D_MASK_POP_D DWC_U2UB_DBGIO_D_MASK_POP_D_W=1	Signal to mask the pop signal in the host FIFO
74	pop_data_in_wait	pop_data_in_wait	host_clk	DWC_U2UB_DBGIO_POP_D_DATA_IN_WAIT DWC_U2UB_DBGIO_POP_D_DATA_IN_WAIT_W=1	Signal to pop the host FIFO data in wait state
77:75	dev_tx_fsm_cs[2:0]	dev_tx_fsm_cs	dev_clk	DWC_U2UB_DBGIO_DTX_FSM_CS DWC_U2UB_DBGIO_DTX_FSM_CS_W=3	Device TX pkt-emulator fms-cs signal
79:78	dev_tx_linestate_sync_eop[1:0]	dev_tx_linestate_sync_eop	dev_clk	DWC_U2UB_DBGIO_DTX_LS_SYNC_EOP DWC_U2UB_DBGIO_DTX_LS_SYNC_EOP_W=2	Dev TX pkt-emulator linestate signal in the sync-eop phase

**Table J-9 Description of Signals in u2u\_soc\_debug\_out (Continued)**

<b>Bit Position</b>	<b>Output Port Name</b>	<b>Internal Signal Name</b>	<b>Clock Domain</b>	<b>Define Name for Bit Position and Width</b>	<b>Description</b>
82:80	host_rx_fsm_cs[2:0]	host_rx_fsm_cs	host_clk	DWC_U2UB_DBGIO_HRX_FSM_CS DWC_U2UB_DBGIO_HRX_FSM_CS_W=3	Host RX pkt-emulator fsm-cs signal
84:83	host_rx_linestate_sync_eop[1:0]	host_rx_linestate_sync_eop	host_clk	DWC_U2UB_DBGIO_HRX_LS_SYNC_EOP DWC_U2UB_DBGIO_HRX_LS_SYNC_EOP_W=2	Host RX pkt-emulator linestate signal during sync-eop phase
85	host_rx_linestate_sync_eop_valid	host_rx_linestate_sync_eop_valid	host_clk	DWC_U2UB_DBGIO_HRX_LS_SYNC_EOP_VALID DWC_U2UB_DBGIO_HRX_LS_SYNC_EOP_VALID_W=1	Host RX pkt-emulator linestate valid signal
86	host_priority	host_priority	host_clk	DWC_U2UB_DBGIO_HOST_PRIORITY DWC_U2UB_DBGIO_HOST_PRIORITY_W=1	Host priority signal -> used to drive linestate
87	host_se0	host_se0	host_clk	DWC_U2UB_DBGIO_HOST_SE0 DWC_U2UB_DBGIO_HOST_SE0_W=1	used to drive SE0 on the host linestate signal
88	dev_txready_gen_frm_ctr	dev_txready_gen_frm_ctr	dev_clk	DWC_U2UB_DBGIO_DTXREADY_GEN_FRM_CTR DWC_U2UB_DBGIO_DTXREADY_GEN_FRM_CTR_W=1	Signal used to drive tx_ready signal during device remote wakeup
92:89	host_tx_sync_cnt[3:0]	sync_cnt	host_clk	DWC_U2UB_DBGIO_HTX_PE_SYNC_CNT DWC_U2UB_DBGIO_HTX_PE_SYNC_CNT_W=4	Host TX pkt-emulator sync-cnt signal
95:93	host_tx_data_cnt[2:0]	data_cnt	host_clk	DWC_U2UB_DBGIO_HTX_PE_DATA_CNT DWC_U2UB_DBGIO_HTX_PE_DATA_CNT_W=3	Host TX pkt-emulator data-cnt signal
97:96	host_tx_eop_cnt[1:0]	eop_cnt	host_clk	DWC_U2UB_DBGIO_HTX_PE_EOP_CNT DWC_U2UB_DBGIO_HTX_PE_EOP_CNT_W=2	Host TX pkt-emulator eop-cnt singal
101:98	dev_rx_sync_cnt[3:0]	sync_cnt	dev_clk	DWC_U2UB_DBGIO_DRX_PE_SYNC_CNT DWC_U2UB_DBGIO_DRX_PE_SYNC_CNT_W=4	Dev RX pkt-emulator sync-cnt signal

**Table J-9 Description of Signals in u2u\_soc\_debug\_out (Continued)**

<b>Bit Position</b>	<b>Output Port Name</b>	<b>Internal Signal Name</b>	<b>Clock Domain</b>	<b>Define Name for Bit Position and Width</b>	<b>Description</b>
104:102	dev_rx_data_cnt[2:0]	data_cnt	dev_clk	DWC_U2UB_DBGIO_DRX_PE_DATA_CNT DWC_U2UB_DBGIO_DRX_PE_DATA_CNT_W=3	Dev RX pkt-emulator data-cnt signal
106:105	dev_rx_eop_cnt[1:0]	eop_cnt	dev_clk	DWC_U2UB_DBGIO_DRX_PE_EOP_CNT DWC_U2UB_DBGIO_DRX_PE_EOP_CNT_W=2	Dev RX pkt-emulator eop-cnt signal
110:107	dev_tx_sync_cnt[3:0]	sync_cnt	dev_clk	DWC_U2UB_DBGIO_DTX_PE_SYNC_CNT DWC_U2UB_DBGIO_DTX_PE_SYNC_CNT_W=4	Dev TX pkt-emulator sync-cnt signal
112:111	dev_tx_data_cnt[2:0]	data_cnt	dev_clk	DWC_U2UB_DBGIO_DTX_PE_DATA_CNT DWC_U2UB_DBGIO_DTX_PE_DATA_CNT_W=2	Dev TX pkt-emulator data-cnt signal
114:113	dev_tx_eop_cnt[1:0]	eop_cnt	dev_clk	DWC_U2UB_DBGIO_DTX_PE_EOP_CNT DWC_U2UB_DBGIO_DTX_PE_EOP_CNT_W=2	Dev TX pkt-emulator eop-cnt signal
118:115	host_rx_sync_cnt[3:0]	sync_cnt	host_clk	DWC_U2UB_DBGIO_HRX_PE_SYNC_CNT DWC_U2UB_DBGIO_HRX_PE_SYNC_CNT_W=4	Host RX pkt-emulator sync-cnt signal
121:119	host_rx_data_cnt[2:0]	data_cnt	host_clk	DWC_U2UB_DBGIO_HRX_PE_DATA_CNT DWC_U2UB_DBGIO_HRX_PE_DATA_CNT_W=3	Host RX pkt-emulator data-cnt signal
123:122	host_rx_eop_cnt[1:0]	eop_cnt	host_clk	DWC_U2UB_DBGIO_HRX_PE_EOP_CNT DWC_U2UB_DBGIO_HRX_PE_EOP_CNT_W=2	Host RX pkt-emulator eop-cnt singal

**Table J-10 Description of Signals in soc\_u2u\_debug\_in**

Bit Position	Parameter Name for Bit Position and Width	Input Ports	Synchronized to Destination Clock?	Clock Domain	Description	Default	Programmable Options
0	DWC_U2UB_DRV_AUXCLK_REQ DWC_U2UB_DRV_AUXCLK_REQ_W=1	drive_auxclk_req	No	asynchronous	If this signal is set, then the DWC_U2UB_DRV_AUXCLK_REQ_VAL value is driven on the auxclk_req output of the U2UB	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/normal operation</li> <li>■ 1-drive_auxclk_req_value is used internally and driven on the auxclk_req o/p</li> </ul>
1	DWC_U2UB_DRV_AUXCLK_REQ_VAL DWC_U2UB_DRV_AUXCLK_REQ_VAL_W=1	drive_auxclk_req_value	No	asynchronous		1'b0	
2	DWC_U2UB_DRV_DEV_VBUSVALID DWC_U2UB_DRV_DEV_VBUSVALID_W=1	drive_dev_vbusvalid	No	asynchronous	If this signal is set, then the DWC_U2UB_DRV_DEV_VBUS_VALID_VAL value is driven on the u2u_dev_vbusvalid output of the U2UB	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/normal operation</li> <li>■ 1-drive_dev_vbusvalid_value is driven on the u2u_dev_vbusvalid line.</li> </ul>
3	DWC_U2UB_DRV_DEV_VBUSVALID_VAL DWC_U2UB_DRV_DEV_VBUSVALID_VAL_W=1	drive_dev_vbusvalid_value	No	asynchronous		1'b0	
4	DWC_U2UB_DRV_UTMICLK_REQ_D DWC_U2UB_DRV_UTMICLK_REQ_D_W=1	drive_utmiclk_req_dev	No	asynchronous	If this signal is set, then the DWC_U2UB_DRV_UTMICLK_REQ_D_VAL value is driven on the u2u_utmiclk_req_dev output of the U2UB	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/normal operation</li> <li>■ 1-drive_utmiclk_req_dev_value is used internally and is driven on the utmiclk_req_dev o/p signal</li> </ul>
5	DWC_U2UB_DRV_UTMICLK_REQ_D_VAL DWC_U2UB_DRV_UTMICLK_REQ_D_VAL_W=1	drive_utmiclk_req_dev_value	No	asynchronous		1'b0	

**Table J-10 Description of Signals in soc\_u2u\_debug\_in (Continued)**

Bit Position	Parameter Name for Bit Position and Width	Input Ports	Synchronized to Destination Clock?	Clock Domain	Description	Default	Programmable Options
6	DWC_U2UB_CLR_D_FIFO DWC_U2UB_CLR_D_FIFO_W=1	clr_dev_fifo	Yes	dev_clk	This signal can be used to clear the device TX FIFO.	1'b0	<ul style="list-style-type: none"> <li>■ 0-Device FIFO is not cleared</li> <li>■ 1- Device FIFO is cleared. Note: This signal needs to be asserted for a minimum of 6 clks</li> </ul>
7	DWC_U2UB_DRV_D_LS DWC_U2UB_DRV_D_LS_W=1	drive_dev_lonestate	No	dev_clk	If this signal is set, then the DWC_U2UB_DRV_D_LS_VAL value is driven on the u2u_dev_utmi_lonestate output line.	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal operation</li> <li>■ 1- drive_dev_lonestate_val is used to drive the u2u_dev_utmi_lonestate and is used internally</li> </ul>
9:8	DWC_U2UB_DRV_D_LS_VAL DWC_U2UB_DRV_D_LS_VAL_W=2	drive_dev_lonestate_value[1:0]	No	dev_clk		2'b00	
10	DWC_U2UB_FORCE_D_OPMODE DWC_U2UB_FORCE_D_OPMODE_W=1	force_dev_opmode	No	dev_clk	If this signal is set, then the DWC_U2UB_FORCE_D_OPMODE_VAL value is driven on the internal u2u_dev_utmi_opmode input of the U2UB	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal operation</li> <li>■ 1-force_dev_opmode_value is driven on the dev_u2u_utmi_opmode line and is used internally.</li> </ul>
12:11	DWC_U2UB_FORCE_D_OPMODE_VAL DWC_U2UB_FORCE_D_OPMODE_VAL_W=2	force_dev_opmode_value[1:0]	No	dev_clk		2'b00	

**Table J-10 Description of Signals in soc\_u2u\_debug\_in (Continued)**

Bit Position	Parameter Name for Bit Position and Width	Input Ports	Synchronized to Destination Clock?	Clock Domain	Description	Default	Programmable Options
13	DWC_U2UB_FORCE_D_XCVRSEL DWC_U2UB_FORCE_D_XCVRSEL_W=1	force_dev_xcvrsel	No	dev_clk	if this signal is set, the the DWC_U2UB_FORCE_D_XCVRSEL_VAL value is driven on the internal u2u_dev_utmi_xcvrselect input of the U2UB	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal operation</li> <li>■ 1-force_dev_xcvrsel_value is driven on the dev_u2u_utmi_xcvrselect line and is used internally.</li> </ul>
15:14	DWC_U2UB_FORCE_D_XCVRSEL_VAL DWC_U2UB_FORCE_D_XCVRSEL_W=2	force_dev_xcvrsel_value[1:0]	No			2'b00	
16	DWC_U2UB_DRV_GATE_D_CLK DWC_U2UB_DRV_GATE_D_CLK_W=1	drive_gate_dev_clk	No	dev_clk	If this signal is set then the DWC_U2UB_DRV_GATE_D_CLK_VAL value is used to gate the u2u_dev_utmi_clk.	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal operation</li> <li>■ 1-drive_gate_dev_clk_value is used to gate the u2u_dev_utmi_clk signal except when dev_u2u_utmi_reset is asserted ( in which case auxclk is selected by default)</li> </ul>
17	DWC_U2UB_DRV_GATE_D_CLK_VAL DWC_U2UB_DRV_GATE_D_CLK_VAL_W=1	drive_gate_dev_clk_value	No			1'b0	

**Table J-10 Description of Signals in soc\_u2u\_debug\_in (Continued)**

Bit Position	Parameter Name for Bit Position and Width	Input Ports	Synchronized to Destination Clock?	Clock Domain	Description	Default	Programmable Options
18	DWC_U2UB_DRV_D_CLK_MUX_SEL DWC_U2UB_DRV_D_CLK_MUX_SEL_W=1	drive_dev_clk_mux_sel	No	dev_clk	If this signal is set, then the DWC_U2UB_DRV_D_CLK_MUX_SEL_VAL value is used as the clock mux select for the internal dev_clk.	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal operation</li> <li>■ 1-When drive_dev_clk_mux_sel_value is 1'b0, utmiclk_dev is used internally, else auxclk is used as the internal dev_clk</li> </ul>
19	DWC_U2UB_DRV_D_CLK_MUX_SEL_VAL DWC_U2UB_DRV_D_CLK_MUX_SEL_VAL_W=1	drive_dev_clk_mux_sel_value	No	dev_clk		1'b0	
20	DWC_U2UB_ASRT_D_U2U_UTMI_RST DWC_U2UB_ASRT_D_U2U_UTMI_RST_W=1	assert_dev_u2u_utmi_reset	No	asynchronous	This signal can be used to assert the device reset.	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal Operation</li> <li>■ 1- The device domain within the bridge is reset.</li> </ul>
21	DWC_U2UB_DRV_UTMICLK_REQ_H DWC_U2UB_DRV_UTMICLK_REQ_H_W=1	drive_utmiclk_req_host	No	asynchronous	If this signal is set, then the DWC_U2UB_DRV_UTMICLK_REQ_H_VAL value is driven on the u2u_utmiclk_req_host output of the U2UB	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/normal operatoin</li> <li>■ 1-drive_utmiclk_req_host_value is used internally and is driven on the utmiclk_req_host o/p signal</li> </ul>
22	DWC_U2UB_DRV_UTMICLK_REQ_H_VAL DWC_U2UB_DRV_UTMICLK_REQ_H_VAL_W=1	drive_utmiclk_req_host_value	No	asynchronous		1'b0	
23	DWC_U2UB_CLR_H_FIFO DWC_U2UB_CLR_H_FIFO_W=1	clr_host_fifo	Yes	host_clk	This signal can be used to clear the host TX FIFO.	1'b0	<ul style="list-style-type: none"> <li>■ 0-Host FIFO is not cleared</li> <li>■ 1- Host FIFO is cleared. Note: This signal needs to be asserted for a minimum of 6 clks</li> </ul>

**Table J-10 Description of Signals in soc\_u2u\_debug\_in (Continued)**

Bit Position	Parameter Name for Bit Position and Width	Input Ports	Synchronized to Destination Clock?	Clock Domain	Description	Default	Programmable Options
24	DWC_U2UB_DRV_H_LS DWC_U2UB_DRV_H_LS_W=1	drive_host_lonestate	No	host_clk	If this signal is set, then the DWC_U2UB_DRV_H_LS_VAL value is driven on the u2u_host_utmi_lonestate output of U2UB	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal operation</li> <li>■ 1- drive_dev_lonestate_val is used to drive the u2u_dev_utmi_lonestate and is used internally</li> </ul>
26:25	DWC_U2UB_DRV_H_LS_VAL DWC_U2UB_DRV_H_LS_VAL_W=2	drive_host_lonestate_value[1:0]	No	host_clk		2'b00	
27	DWC_U2UB_FORCE_H_OPMODE DWC_U2UB_FORCE_H_OPMODE_W=1	force_host_opmode	No	host_clk	If this signal is set, then the DWC_U2UB_FORCE_H_OPMODE_VAL value is driven on the internal u2u_host_utmi_opmode input of the U2UB	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal operation</li> <li>■ 1-force_host_opmode_value is driven on the host_u2u_utmi_opmode line and is used internally.</li> </ul>
29:28	DWC_U2UB_FORCE_H_OPMODE_VAL 'DWC_U2UB_FORCE_H_OPMODE_VAL_W=2	force_host_opmode_value[1:0]	No	host_clk		2'b00	
30	DWC_U2UB_FORCE_H_XCVRSEL DWC_U2UB_FORCE_H_XCVRSEL_W=1	force_host_xcvrsel	No	host_clk	If this signal is set, then the DWC_U2UB_FORCE_H_XCVRSEL_VAL value is driven on the internal u2u_host_utmi_xcvrselect input of the U2UB	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal operation</li> <li>■ 1-force_host_xcvrsel_value is driven on the host_u2u_utmi_xcvrselect line and is used internally.</li> </ul>
32:31	DWC_U2UB_FORCE_H_XCVRSEL_VAL 'DWC_U2UB_FORCE_H_XCVRSEL_VAL_W=2	force_host_xcvrsel_value[1:0]	No	host_clk		2'b00	

**Table J-10 Description of Signals in soc\_u2u\_debug\_in (Continued)**

Bit Position	Parameter Name for Bit Position and Width	Input Ports	Synchronized to Destination Clock?	Clock Domain	Description	Default	Programmable Options
33	DWC_U2UB_DRV_GATE_H_CLK DWC_U2UB_DRV_GATE_H_CLK_W=1	drive_gate_host_clk	No	host_clk	If this signal is set then the DWC_U2UB_DRV_GATE_H_CLK_VAL value is used to gate the u2u_host_utmi_clk.	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal operation</li> <li>■ 1-drive_gate_host_clk_value is used to gate the u2u_host_utmi_clk signal except when host_u2u_utmi_reset is asserted ( in which case auxclk is selected by default)</li> </ul>
34	DWC_U2UB_DRV_GATE_H_CLK_VAL DWC_U2UB_DRV_GATE_H_CLK_VAL_W=1	drive_gate_host_clk_value	No	host_clk		1'b0	
35	DWC_U2UB_DRV_H_CLK_MUX_SEL DWC_U2UB_DRV_H_CLK_MUX_SEL_W=1	drive_host_clk_mux_sel	No	host_clk	If this signal is set, then the DWC_U2UB_DRV_H_CLK_MUX_SEL_VAL value is used as the clock mux select for the internal host_clk.	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal operation</li> <li>■ 1-When drive_host_clk_mux_sel_value is 1'b0, utmiclk_host is used internally, else auxclk is used as the internal dev_clk</li> </ul>
36	DWC_U2UB_DRV_H_CLK_MUX_SEL_VAL DWC_U2UB_DRV_H_CLK_MUX_SEL_VAL_W=1	drive_host_clk_mux_sel_value	No	host_clk		1'b0	
37	DWC_U2UB_ASRT_H_U2U_UTMI_RST DWC_U2UB_ASRT_H_U2U_UTMI_RST_W=1	assert_host_u2u_utmi_reset	No	asynchronous	This signal can be used to assert the host reset.	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal Operation</li> <li>■ 1- The host domain within the bridge is reset.</li> </ul>

**Table J-10 Description of Signals in soc\_u2u\_debug\_in (Continued)**

Bit Position	Parameter Name for Bit Position and Width	Input Ports	Synchronized to Destination Clock?	Clock Domain	Description	Default	Programmable Options
38	DWC_U2UB_DRVHTXPE_UTMI_TXVALID DWC_U2UB_DRVHTXPE_UTMI_TXVALID_W=1	drive_htx_pe_utmi_txvalid	No	host_clk	This signal can be used to change the FSM state from IDLE-SYNC or DATA->EOP	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal Operation</li> <li>■ 1-drive_htx_pe_utmi_txvalid_val is driven as the control signal for the FSMin the htx pkt emulator</li> </ul>
39	DWC_U2UB_DRVHTXPE_UTMI_TXVALID_VAL DWC_U2UB_DRVHTXPE_UTMI_TXVALID_VAL_W=1	drive_htx_pe_utmi_txvalid_val		host_clk		1'b0	
40	DWC_U2UB_DRVDRXPE_RX_SYNC_EOP DWC_U2UB_DRVDRXPE_RX_SYNC_EOP_W=1	drive_drx_pe_rx_sync_eop		dev_clk	This signal can be used to change the FSM state from IDLE-SYNC or DATA->EOP	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal Operation</li> <li>■ 1-drive_drx_pe_rx_sync_eop_val is used as the control signal for the dev-rx pkt emulator fsm</li> </ul>
41	DWC_U2UB_DRVDRXPE_RX_SYNC_EOP_VAL DWC_U2UB_DRVDRXPE_RX_SYNC_EOP_VAL_W=1	drive_drx_pe_rx_sync_eop_val		dev_clk		1'b0	
42	DWC_U2UB_DRVDTXPE_UTMI_TXVALID DWC_U2UB_DRVDTXPE_UTMI_TXVALID_W=1	drive_dtx_pe_utmi_txvalid	No	dev_clk	This signal can be used to change the FSM state from IDLE-SYNC or DATA->EOP	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal Operation</li> <li>■ 1-drive_dtx_pe_utmi_txvalid_val is driven as the control signal for the FSMin the dtx pkt emulator</li> </ul>
43	DWC_U2UB_DRVDTXPE_UTMI_TXVALID_VAL DWC_U2UB_DRVDTXPE_UTMI_TXVALID_VAL_W=1	drive_dtx_pe_utmi_txvalid_val	No	dev_clk		1'b0	

**Table J-10 Description of Signals in soc\_u2u\_debug\_in (Continued)**

Bit Position	Parameter Name for Bit Position and Width	Input Ports	Synchronized to Destination Clock?	Clock Domain	Description	Default	Programmable Options
44	DWC_U2UB_DRV_HRX_PE_RX_SYNC_EOP DWC_U2UB_DRV_HRX_PE_RX_SYNC_EOP_W=1	drive_hrx_pe_rx_sync_eop		host_clk	This signal can be used to change the FSM state from IDLE-SYNC or DATA->EOP	1'b0	<ul style="list-style-type: none"> <li>■ 0-Inactive/Normal Operation</li> <li>■ 1-drive_hrx_pe_rx_sync_eop_val is used as the control signal for the host-rx pkt emulator fsm</li> </ul>
45	DWC_U2UB_DRV_HRX_PE_RX_SYNC_EOP_VAL DWC_U2UB_DRV_HRX_PE_RX_SYNC_EOP_VAL_W=1	drive_hrx_pe_rx_sync_eop_val		host_clk		1'b0	
46	DWC_U2UB_DRV_DBYP_CLKRDY_GATE DWC_U2UB_DRV_DBYP_CLKRDY_GATE_W=1	drive_dev_byp_clk_ready_gate_en		dev_clk	-	1'b0	<ul style="list-style-type: none"> <li>■ 0- Inactive/Normal Operation</li> <li>■ 1-drive_dev_byp_clk_ready_gate_en is used to bypass the gating the u2u_dev_utmi_clk when it switches from utmiclk_dev to aux_clk during the assertion of *dev*suspend_n. u2u_dev_utmi_clk will be driven for a short duration to aux_clk during this transition period.</li> </ul> <p><b>Note:</b> Synopsys does not recommend setting this debug port to 1'b1.</p>

**Table J-10 Description of Signals in soc\_u2u\_debug\_in (Continued)**

Bit Position	Parameter Name for Bit Position and Width	Input Ports	Synchronized to Destination Clock?	Clock Domain	Description	Default	Programmable Options
47	DWC_U2UB_DRV_HBYP_CLKRDY_GATE DWC_U2UB_DRV_HBYP_CLKRDY_GATE_W=1	drive_host_byp_clk_ready_gate_en		host_clk	-	1'b0	<ul style="list-style-type: none"> <li>■ 0- Inactive/Normal Operation</li> <li>■ 1-drive_host_byp_clk_ready_gate_en is used to bypass the gating the u2u_host_utmi_clk when it switches from utmiclk_host to aux_clk during the assertion of *host*_suspend_n. u2u_host_utmi_clk will be driven for a short duration to aux_clk during this transition period.</li> </ul> <p><b>Note:</b> Synopsys does not recommend setting this debug port to 1'b1.</p>

### J.6.5 16-Bit Data Interface

The U2U Bridge provides support 16-bit parallel data interface. This is a configurable option by setting the DWC\_U2UB\_DATA\_BUS\_WIDTH to 16 instead of the default value of 8.

When 16-bit parallel data interface is enabled the following UTMI+ signals will be used/driven (on both the host and the device interfaces):

- utmi\_txdata[15:8], utmi\_txvalidh
- utmi\_rxdata[15:8], utmi\_rxvalidh

The frequency of operation is 30MHz. The SoC must drive 30MHz clock on the respective utmi\_clk lines (instead of 60MHz). There is no need to change the auxclk frequency. The operation of the U2U Bridge is as specified by the *UTMI+ Specification*.



**Note** The DWC\_U2UB\_DATA\_BUS\_WIDTH parameter is set to 8bit or 16bit automatically based on the UTMI Data Width Configuration Parameter chosen in the DWC\_otg controller.

## J.7 Registers

There are no programmable registers in the U2U Bridge controller. The U2U Bridge requires no change at the driver level in the host and device controllers.

## J.8 Area

Table J-11 shows the area of U2U Bridge for configurations based on Data Width.

**Table J-11 U2U Bridge – Area**

Configuration	Area
DWC_U2UB_DATA_BUS_WIDTH = 8	7.502K
DWC_U2UB_DATA_BUS_WIDTH = 16	10.006K



**Note** The area numbers are in Kilo-gates (K = 1,000 gates) for a well-known 0.065 µm technology. The total cell area is divided by the area of the smallest NAND cell in the library to get the area numbers.

## J.9 References

[Table J-12](#) lists the references.

**Table J-12 References**

Document Name	Revision No.	Date
Universal Serial Bus Revision 2.0 specification (.zip format. Enclosed in this zip file are the following documents: ) <ul style="list-style-type: none"> <li>■ The Original USB 2.0 specification released on April 27, 2000</li> <li>■ Errata to the USB 2.0 specification as of December 7, 2000</li> <li>■ Mini-B connector Engineering Change Notice to the USB 2.0 specification.</li> <li>■ Pull-up/pull-down Resistors Engineering Change Notice to the USB 2.0 specification.</li> <li>■ Errata to the USB 2.0 specification as of May 28, 2002</li> <li>■ Interface Association Descriptor Engineering Change Notice to the USB 2.0 specification.</li> <li>■ Rounded Chamfer Engineering Change Notice to the USB 2.0 specification as of October 8, 2003</li> <li>■ Unicode Engineering Change Notice to the USB 2.0 specification as of February 21, 2005</li> <li>■ Inter-Chip USB Supplement Revision 1.0 as of March 13, 2006</li> <li>■ Revision 1.3 of the USB On-The-Go Supplement as of December 5, 2006</li> <li>■ Revision 1.01 of the Micro-USB Cables and Connectors Specification as of April 4, 2007</li> <li>■ USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification as of July 16, 2007.</li> </ul>	-	August 11, 2014
USB 2.0 Transceiver Macrocell Interface (UTMI) Specification	Ver 1.05	March 29. 2001
UTMI+ Specification	Rev 1.0	Feb 25, 2004

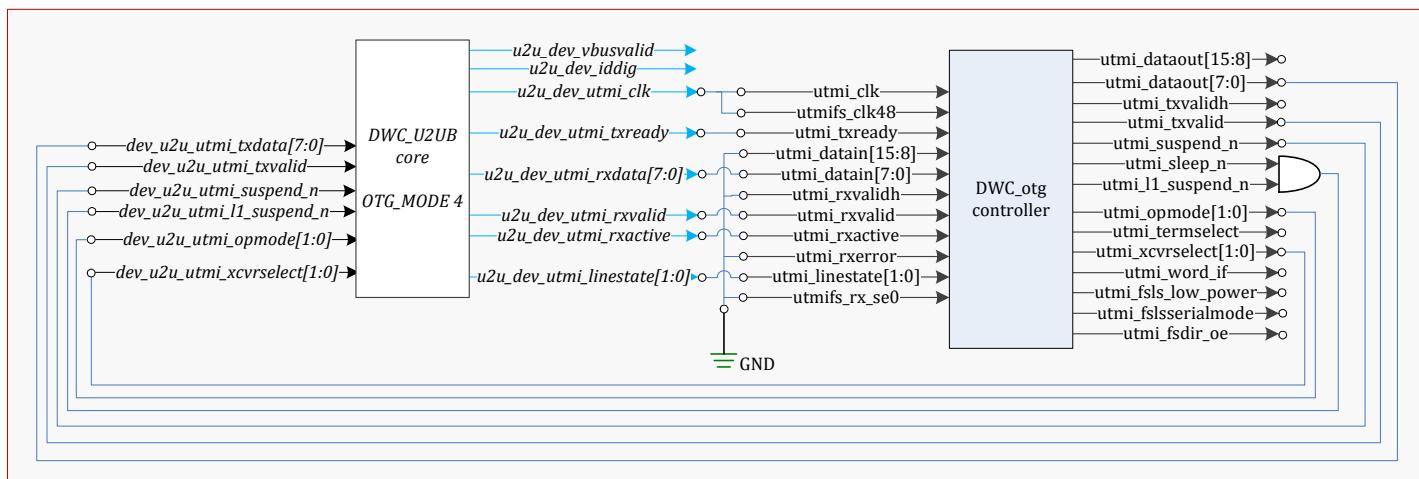
## J.10 Integrating DWC\_U2UB Device Interface with DWC\_otg Controller in Non-OTG Device Mode

**Figure J-10** shows example interconnections between the DWC\_U2UB and the DWC\_otg controller in non-OTG device mode (OTG\_MODE = 4). The DWC\_otg controller does not require u2u\_dev\_vbusvalid and u2u\_dev\_iddig outputs from the DWC\_U2UB controller when OTG\_MODE = 4.

The DWC\_U2UB controller,

- Ignores utmi\_word\_if (8-bit assumed)  
When the bridge data interface is 8-bit wide, it ignores the upper byte of utmi\_txdata/utmi\_rxdata (along with corresponding txvalidh/rxvalidh)
- Does not use any of the fsls\_serialmode signals
- Does not use utmi\_termselect
- Does not drive utmi\_rxerror, utmi\_rxvalidh, utmi\_datain[15:8], and utmifs\_rx\_se0. These signals must be tied to GND
- Does not provide 48 MHz clock, therefore, utmifs\_clk48 needs to be shorted to utmi\_clk
- Treats assertion of utmi\_l1\_suspend\_n equivalent to the assertion of utmi\_sleep\_n

**Figure J-10 Example Interconnection Between DWC\_U2UB and a Non-OTG Device**



For details on creating an example DWC\_otg DWC\_U2UB subsystem using coreAssembler, refer to the Section 2.15 in the *DesignWare Cores USB 2.0 Hi-Speed On-The-Go (OTG) User Guide*.



# K

## Chirp\_on Behavior

---

The [chirp\\_on](#) signal can be used to know about an imminent Chirp 'K' signaling when DWC\_otg is acting as a device. This requirement is from the Battery Charging standard which mandates that during Chirp 'K', the device must lower its current consumption from IDEV\_HCHG\_HS (900 mA) to IDEV\_HCHG\_CHRP (560 mA).

This appendix describes the behavior of the [chirp\\_on](#) signal in the following scenarios:

- “FS Idle to Host Reset”
- “FS Suspend to Host Reset” on page [832](#)
- “HS Idle to Host Reset” on page [833](#)
- “HS Idle to Host Suspend to Host Reset” on page [834](#)
- “HS IDLE to HS LPM Suspend to Host Reset” on page [835](#)



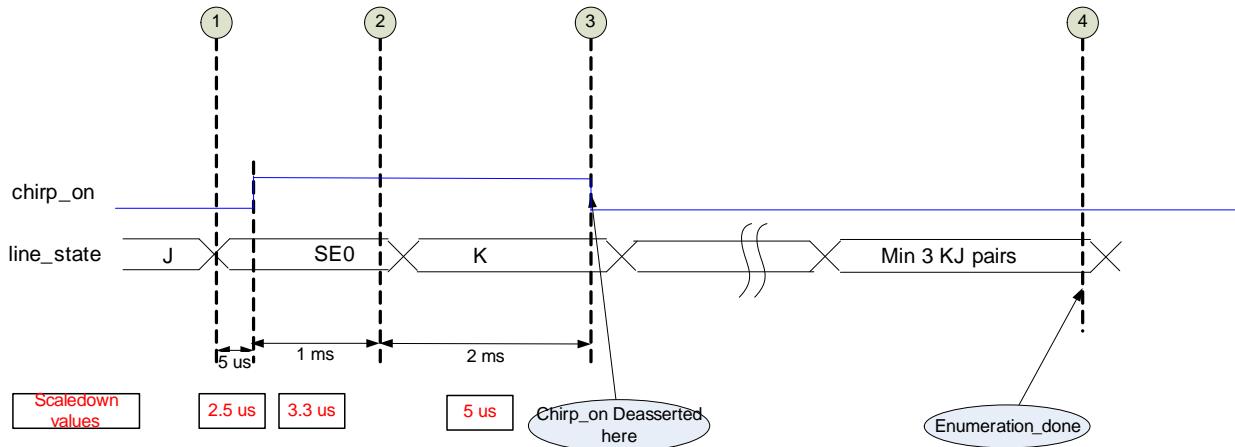
**Note** The timing diagrams are not to scale and are illustrative only. In all these scenarios, DWC\_otg is a USB device.

---

## K.1 FS Idle to Host Reset

In this scenario, USB is in Full Speed Idle state (J) when the USB Host drives a SE0 to signal a USB reset.

**Figure K-1 Chirp\_on Behavior During FS Idle to Host Reset**



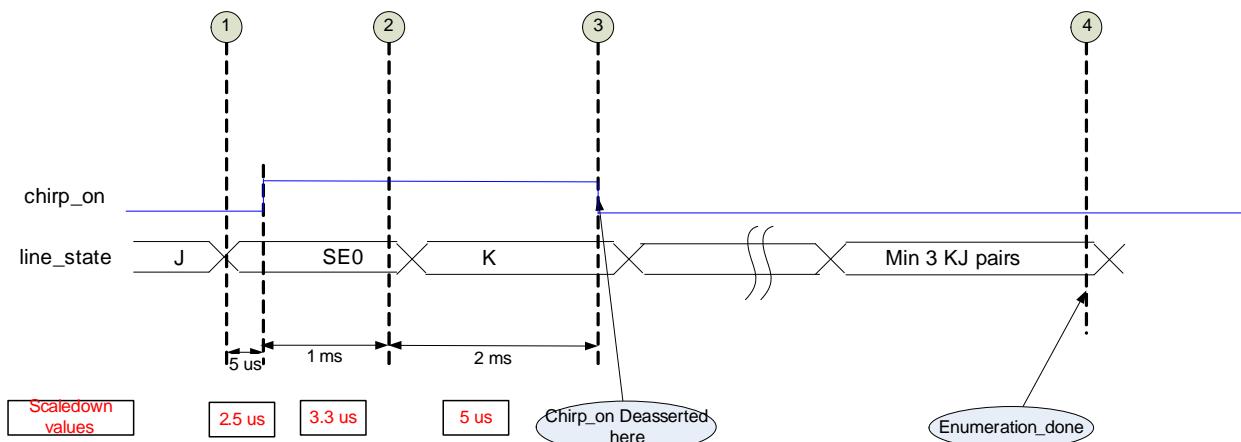
1. The USB host drives a SE0 to indicate a USB reset. 5  $\mu$ s ( $\pm 1 \mu$ s) after sensing SE0, DWC\_otg asserts chirp\_on.
2. 1 ms ( $\pm 1 \mu$ s) after asserting chirp\_on, DWC\_otg drives a Chirp K on the USB. The duration of the Chirp K is 2 ms ( $\pm 1 \mu$ s).
3. DWC\_otg stops driving Chirp K on USB along with de-assertion of chirp\_on.
4. As a consequence of step 3, the USB host may respond with a minimum of 3 KJ pairs, completing the High Speed reset protocol. In this case, the device enumerates as High speed or the host may also not respond with Chirp in which case the device enumerates as Full Speed.



The  $\pm 1 \mu$ s timing variance is provided for keeping the design simple and is applicable to all scenarios described in this appendix.

## K.2 FS Suspend to Host Reset

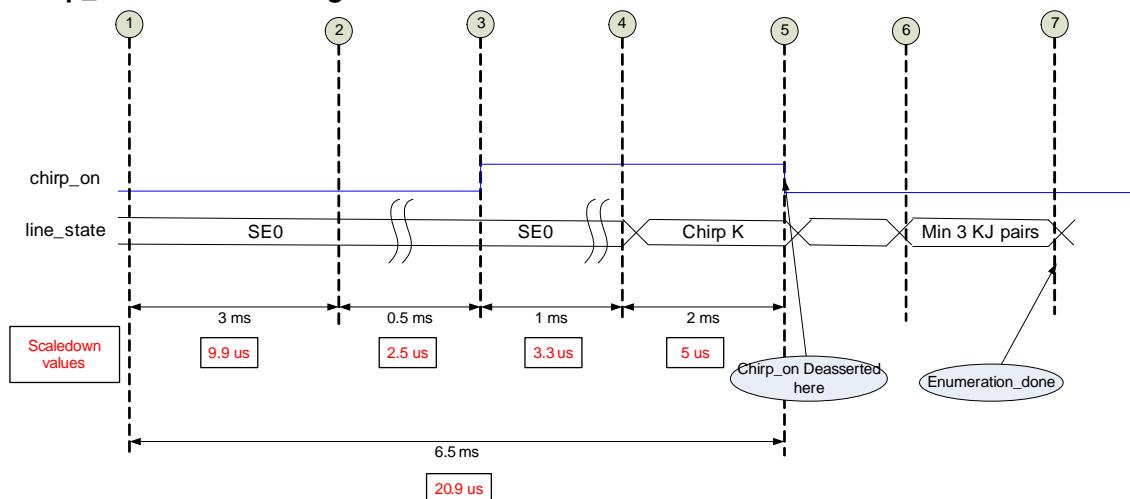
In this scenario, USB is in Full Speed Suspend state when the USB Host drives a SE0 to signal a USB reset.

**Figure K-2 Chirp\_on Behavior During FS Suspend to Host Reset**

1. The USB host drives a SE0 to indicate a USB reset. 5 us after sensing SE0, DWC\_otg drives chirp\_on.
2. 1 ms after asserting chirp\_on, DWC\_otg drives a Chirp K on the USB. The duration of the Chirp K is 2 ms.
3. DWC\_otg stops driving Chirp K on USB along with de-assertion of chirp\_on.
4. As a consequence of step 3, the USB host may respond with a minimum of 3 KJ pairs, completing the High Speed reset protocol. In this case, the device enumerates as High speed or the host may also not respond with Chirp in which case the device enumerates as Full Speed.

### K.3 HS Idle to Host Reset

HS Idle state on USB is a SE0 for =< 3 ms. Beyond this time, the USB Host goes into Suspend state when DWC\_otg drives a 'J' state. If the Host is driving SE0 even after 3.5 ms, DWC\_otg interprets this as a Host Reset and drives a Chirp 'K'.

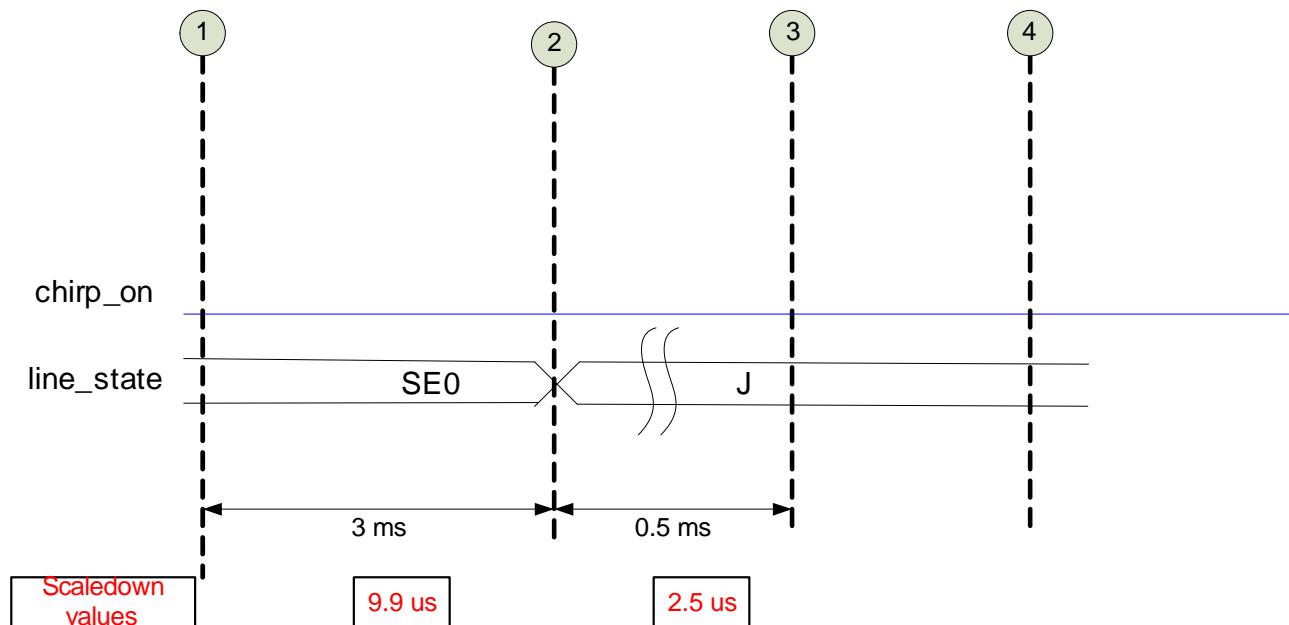
**Figure K-3 Chirp\_on Behavior During HS Idle to Host Reset**

1. The USB host drives a SE0 to indicate a USB reset. (At this point of time, DWC\_otg is unable to determine if the SE0 on USB is due to USB host driving a SE0 or due to inactivity on USB).

2. There is no activity on USB for 3 ms. DWC\_otg pulls up the D+ line to indicate a J on USB. Because the host is driving a strong SE0, it does not get reflected on USB. DWC\_otg keeps its D+ line pulled up for 0.5 ms.
3. DWC\_otg dssr\_state checks for the USB line state and finds that it is still SE0. Because of this, chirp\_on is asserted. At the same time, a 1 ms timer is started by DWC\_otg.
4. At the end of 1 ms timeout, Chirp 'K' is driven by DWC\_otg for a duration of 2 ms.
5. At the end of 2 ms duration of driving chirp 'K', DWC\_otg deasserts chirp\_on signal.
6. The USB host (if it is HS-capable) recognizes the Chirp 'K' and start to respond with minimum of 3 KJ pairs.
7. The HS/FS enumeration process is complete.

## K.4 HS Idle to Host Suspend to Host Reset

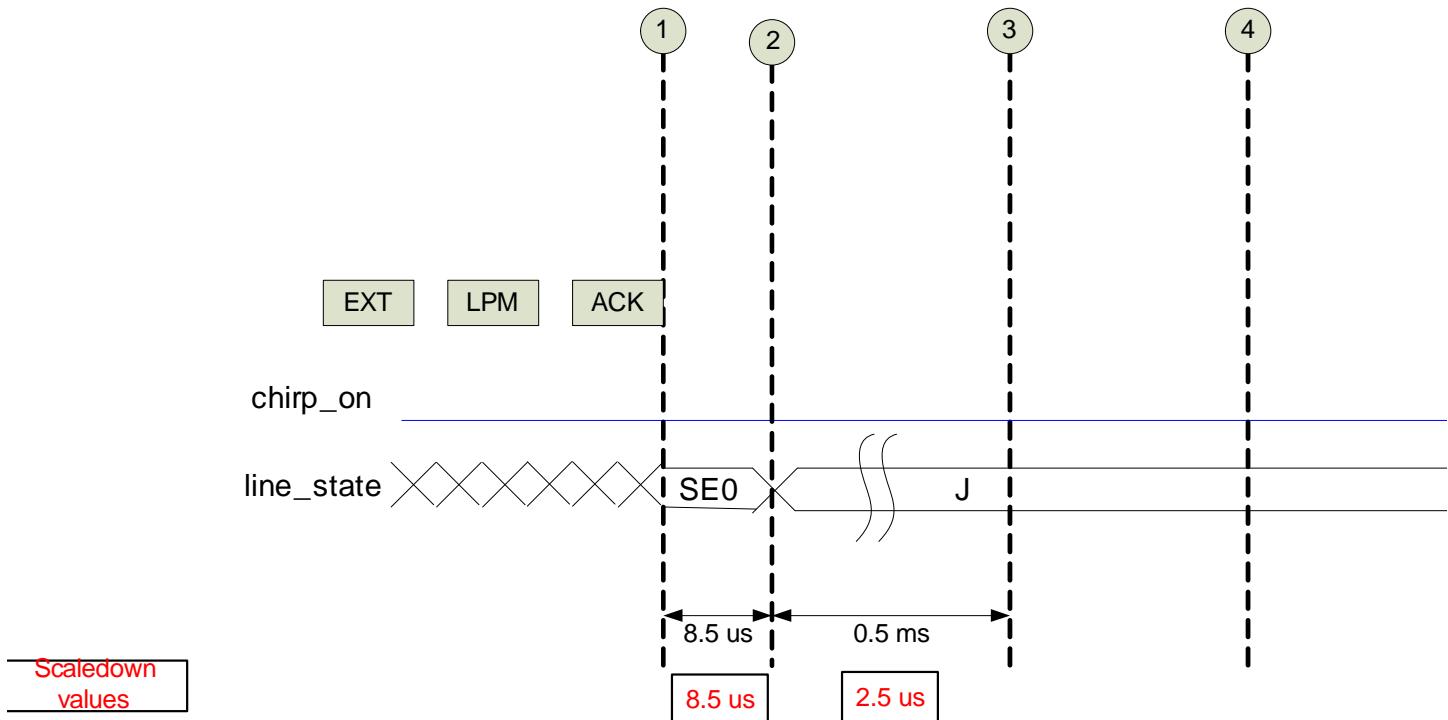
**Figure K-4 Chirp\_on Behavior During HS Idle to Host Suspend**



1. There is no activity on USB. (At this point of time, DWC\_otg is unable to determine if the SE0 on USB is due to USB host driving a SE0 or due to inactivity on USB).
2. There is no activity on USB for 3 ms. DWC\_otg pulls up the D+ line to indicate a J on USB. Because the host is not driving a strong SE0, it gets reflected on USB as a J state. DWC\_otg keeps its D+ line pulled up for 0.5 ms.
3. DWC\_otg checks for the USB line state and finds that it is a J state. Because of this, chirp\_on is not asserted.
4. Because of 'J' state on USB, DWC\_otg goes to suspend state.
5. DWC\_otg then follows the steps listed in “[FS Suspend to Host Reset](#)” on page 832.

## K.5 HS IDLE to HS LPM Suspend to Host Reset

**Figure K-5 Chirp\_on Behavior for HS Idle to Suspend During LPM**



1. There is no activity on USB. Prior to this event, an LPM token is acknowledged.
2. There is no activity on USB for 8.5 us. DWC\_otg pulls up the D+ line to indicate a J on USB. Since the host is not driving a strong SE0, it gets reflected on USB as a J state. DWC\_otg keeps its D+ line pulled up for 0.5 ms.
3. DWC\_otg checks for the USB line state and finds that it is a J state. Because of this, chirp\_on is not asserted.
4. Because of 'J' state on USB, DWC\_otg goes to suspend state.
5. DWC\_otg then follows the steps listed in "["FS Suspend to Host Reset"](#) on page 832.



## L

# Internal Parameter Descriptions

Provides a description of the internal parameters that might be indirectly referenced in expressions in the Signals, Parameters, or Registers chapters. These parameters are not visible in the coreConsultant GUI and most of them are derived automatically from visible parameters. **You must not set any of these parameters directly.**

Some expressions might refer to TCL functions or procedures (sometimes identified as **function\_of**) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the core in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

**Table L-1 Internal Parameters**

Parameter Name	Equals To
DAINT	6
DAINTMSK	7
DCFG	0
DCTL	1
DEACHINT	14
DEACHINTMSK	15
DIEPEACHMSK0	16
DIEPMSK	4
DOEPEACHMSK0	32
DOEPMSK	5
DSTS	2
DTHRCTL	12
DTKNQR1	8

**Table L-1 Internal Parameters (Continued)**

Parameter Name	Equals To
DTKNQR2	9
DTKNQR3	12
DTKNQR4	13
DVBUSDIS	10
DVBUSPULSE	11
EP_LOC_CNT	128
FS	2'b01
FS48	2'b11
FS_TERM	1'b1
FS_XCVR	2'b01
GAHBCFG	2
GDFIFO CFG	23
GGPIO	14
GHWCFG1	17
GHWCFG2	18
GHWCFG3	19
GHWCFG4	20
GI2CCTL	12
GINTMSK	6
GINTMSK2	26
GINTSTS	5
GINTSTS2	27
GLPMCFG	21
GNPTXFSIZ	10
GNPTXSTS	11
GOTGCTL	0
GOTGINT	1
GPVNDCTL	13

**Table L-1 Internal Parameters (Continued)**

Parameter Name	Equals To
GPWRDN	22
GREFCLK	25
GRSTCTL	4
GRXFSIZ	9
GRXSTSP	8
GRXSTS	7
GSNPSID	16
GUID	15
GUSBCFG	3
H	1'b1
HAINT	5
HAINTMSK	6
HCFG	0
HFIR	1
HFNUM	2
HPTXFSIZ	3
HPTXSTS	4
HS	2'b00
HS_TERM	1'b0
HS_XCVR	2'b00
LFS_XCVR	2'b11
LS	2'b10
LS_XCVR	2'b10
OTG_ADC30_REF_CLK_EN	{(OTG_SERV_INT_ENH ==1 ? 1:0)}
OTG_DF_AWID	{[ ::DWC_otg::otg_log2 OTG_DFIFO_DEPTH ]}
OTG_ISOC_IPG_ENH	{((OTG_MODE <= 4) && (OTG_HSPHY_INTERFACE !=0))}
OTG_PWR_CLAMP	(OTG_EN_PWROPT !=0 && OTG_EN_PWROPT !=3)
OTG_TYPEC_MODE	= 0

**Table L-1 Internal Parameters (Continued)**

Parameter Name	Equals To
PHY_INST	{[::DWC_otg::Check_SE_CORE_hidden]}