



DesignWare Cores DWC MIPI I3C Master and Slave Controller

Databook

DWC MIPI I3C CONTROLLER - Product Code: B611-0

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

Contents

Revision History	7
Preface	13
Chapter 1	
Product Overview	17
1.1 General Product Description	18
1.1.1 System Block Diagram	18
1.1.2 Applications	19
1.1.3 Standards Compliance	20
1.2 DWC_mipi_i3c Features	21
1.2.1 General Features	21
1.2.2 DWC_mipi_i3c Master Features	21
1.2.3 DWC_mipi_i3c APB Slave Features	22
1.2.4 DWC_mipi_i3c Slave-Lite Features	22
1.2.5 Unsupported Protocol Features for Master and Secondary Master	22
1.2.6 Unsupported Protocol Features for APB Slave	23
1.2.7 Interfaces	23
1.2.8 Master (Main Master and Secondary Master) and APB Slave	23
1.2.9 Bus Configuration	24
1.2.10 Device Address	24
1.2.11 In-Band Interrupt (IBI)	25
1.2.12 Operation Modes	25
1.3 Speed and Clock Requirements	26
1.3.1 Speed and Clock requirements of DWC_mipi_i3c Master	26
1.3.2 Speed and Clock requirements of DWC_mipi_i3c APB Slave	26
1.4 Deliverables	26
Chapter 2	
Architecture of the DWC_mipi_i3c	29
2.1 Block Diagram and Description of DWC_mipi_i3c	30
2.1.1 Slave Bus Interface Module	30
2.1.2 Register Block Module	30
2.1.3 Interrupt Control Module	31
2.1.4 DMA Handshake Interface Module	31
2.1.5 Async Queue Block	31
2.1.6 RAM Access Control	31
2.1.7 Data Router Block	31
2.1.8 Master Transaction Control Module	31
2.1.9 Slave Transaction Control Module	31

2.1.10 Transmit Control and Receive Control Blocks	31
2.1.11 Bus Monitor	32
2.2 Description of DWC_mipi_i3c Bus Interfaces	33
2.2.1 I/O Buffer Interface	33
2.2.2 APB Interface	34
2.2.3 Interrupt Interface	34
2.2.4 DMA Handshake Interface	34
2.2.5 Debug Interface	34
2.2.6 Single Port RAM Interface	34
2.3 Simple-Transfer DMA (SDMA) Handshake	35
2.3.1 Overview of SDMA Handshake in DWC_mipi_i3c	35
2.3.2 Description of SDMA Handshake in DWC_mipi_i3c	35
2.3.3 Configuring SDMA in DWC_mipi_i3c	37
2.4 SPRAM and Buffer Depth in DWC_mipi_i3c	38
2.4.1 Overview of SPRAM and Buffer Depth in DWC_mipi_i3c	38
2.4.2 Configuring Buffer Depths	38
2.5 Interrupts in DWC_mipi_i3c	40
2.5.1 Overview of Interrupts in DWC_mipi_i3c	40
2.5.2 Description of Interrupt Mechanism in DWC_mipi_i3c	40
2.6 Master Functionality with DWC_mipi_i3c	42
2.6.1 Overview of Master Role in DWC_mipi_i3c	42
2.6.2 Dynamic Address Assignment (DAA)	42
2.6.3 In-Band Interrupt (IBI) Detection and Handling	47
2.6.4 Disabling DWC_mipi_i3c Master	54
2.6.5 Aborting Transfers of DWC_mipi_i3c Master	54
2.6.6 Master Data Structures in Non-HCI Mode	55
2.6.7 Master Data Structures in HCI Mode	70
2.6.8 Operation Modes of DWC_mipi_i3c	90
2.6.9 SCL Generation and Timings Based on Bus Configuration	123
2.6.10 Derivation of I3C/I2C Timing Parameters from Timing Registers	127
2.6.11 Error Detection	130
2.6.12 Defining Byte Support	132
2.6.13 Packet Error Check	132
2.6.14 Broadcast CCC's in I2C Speed	133
2.6.15 BUS RESET Generation	133
2.7 Slave Functionality with DWC_mipi_i3c	135
2.7.1 Overview of Slave Role in DWC_mipi_i3c	135
2.7.2 Description of the Slave Role in DWC_mipi_i3c	135
2.7.3 I3C vs I2C Role Selection	135
2.7.4 Slave Role Related Registers	136
2.7.5 Handling Address Assignment	137
2.7.6 CCC Transfers with DWC_mipi_i3c Slave	137
2.7.7 Private Data Transfers	144
2.7.8 Handling Private Transmit (Master Read) Transfers	145
2.7.9 Hot-Join Request Generation	146
2.7.10 Slave Interrupt Request Generation	147
2.7.11 Master Request Generation	148
2.7.12 Disabling DWC_mipi_i3c Slave	149
2.7.13 Data Structure in DWC_mipi_i3c Slave	149

Chapter 3

Parameter Descriptions	153
3.1 Role Configuration Parameters	154
3.2 HCI Configuration Parameters	155
3.3 Basic Configuration Parameters	157
3.4 Queues and Interfaces Parameters	158
3.5 Master Configuration Parameters	159
3.6 Slave Configuration Parameters	161
3.7 Clock(s) Configuration Parameters	165
3.8 External Memory Parameters	172
3.9 Preset Values Parameters	173
3.10 HCI Preset Values Parameters	185

Chapter 4

Signal Descriptions	187
4.1 APB Interface Signals	189
4.2 RAM Interface Signals	193
4.3 Core Interface Signals	195
4.4 HDR Interface Signals	196
4.5 Interrupt Interface Signals	198
4.6 SDMA Interface Signals	199
4.7 Debug Interface Signals	202
4.8 I3C Interface Signals	204
4.9 Slave Interface Signals	207
4.10 Scan Interface Signals	211

Chapter 5

Register Descriptions	213
5.1 DWC_mipi_i3c_HCI_map/DWC_mipi_i3c_HCI_block Registers	217
5.1.1 HCI_VERSION	220
5.1.2 HC_CONTROL	221
5.1.3 MASTER_DEVICE_ADDR	225
5.1.4 HC_CAPABILITIES	226
5.1.5 RESET_CONTROL	228
5.1.6 PRESENT_STATE	230
5.1.7 INTR_STATUS	231
5.1.8 INTR_STATUS_ENABLE	232
5.1.9 INTR_SIGNAL_ENABLE	233
5.1.10 INTR_FORCE	234
5.1.11 DAT_SECTION_OFFSET	235
5.1.12 DCT_SECTION_OFFSET	236
5.1.13 RING_HEADERS_SECTION_OFFSET	238
5.1.14 PIO_SECTION_OFFSET	239
5.1.15 EXTCAPS_SECTION_OFFSET	240
5.1.16 IBI_NOTIFY_CTRL	241
5.1.17 DEV_CTX_BASE_LO	243
5.1.18 DEV_CTX_BASE_HI	244
5.1.19 HW_IDENTIFICATION_HEADER	245
5.1.20 COMP_MANUFACTURER	246

5.1.21 COMP_VERSION	247
5.1.22 COMP_TYPE	248
5.1.23 BUS_TIMING_HEADER	249
5.1.24 SCL_I3C_OD_TIMING	250
5.1.25 SCL_I3C_PP_TIMING	251
5.1.26 SCL_I2C_FM_TIMING	252
5.1.27 SCL_I2C_FMP_TIMING	253
5.1.28 SCL_I2C_SS_TIMING	254
5.1.29 SCL_EXT_LCNT_TIMING	255
5.1.30 SCL_EXT_TERMN_LCNT_TIMING	257
5.1.31 SDA_HOLD_SWITCH_DLY_TIMING	259
5.1.32 BUS_FREE_TIMING	261
5.1.33 SCL_LOW_MST_EXT_TIMEOUT	262
5.1.34 DS_EXTCAP_HEADER	263
5.1.35 QUEUE_STATUS_LEVEL	264
5.1.36 DATA_BUFFER_STATUS_LEVEL	266
5.1.37 PRESENT_STATE_DEBUG	267
5.1.38 MASTER_EXT_HEADER	273
5.1.39 MASTER_CONFIG	274
5.1.40 COMMAND_QUEUE_PORT	276
5.1.41 RESPONSE_QUEUE_PORT	277
5.1.42 RX_DATA_PORT	280
5.1.43 TX_DATA_PORT	281
5.1.44 IBI_PORT	282
5.1.45 QUEUE_THLD_CTRL	283
5.1.46 DATA_BUFFER_THLD_CTRL	286
5.1.47 QUEUE_SIZE_CTRL	291
5.1.48 PIO_INTR_STATUS	293
5.1.49 PIO_INTR_STATUS_ENABLE	295
5.1.50 PIO_INTR_SIGNAL_ENABLE	297
5.1.51 PIO_INTR_FORCE	299
5.1.52 DEV_ADDR_TABLE1_LOC1	301
5.1.53 DEV_ADDR_TABLE1_LOC2	304
5.1.54 DEV_CHAR_TABLE1_LOC1	306
5.1.55 DEV_CHAR_TABLE1_LOC2	307
5.1.56 DEV_CHAR_TABLE1_LOC3	308
5.1.57 DEV_CHAR_TABLE1_LOC4	309
5.2 DWC_mipi_i3c_map/DWC_mipi_i3c_block Registers	310
5.2.1 DEVICE_CTRL	313
5.2.2 DEVICE_ADDR	319
5.2.3 HW_CAPABILITY	323
5.2.4 COMMAND_QUEUE_PORT	326
5.2.5 RESPONSE_QUEUE_PORT	327
5.2.6 RX_DATA_PORT	328
5.2.7 TX_DATA_PORT	329
5.2.8 IBI_QUEUE_DATA	330
5.2.9 IBI_QUEUE_STATUS	331
5.2.10 QUEUE_THLD_CTRL	333
5.2.11 DATA_BUFFER_THLD_CTRL	336

5.2.12 IBI_QUEUE_CTRL	340
5.2.13 IBI_MR_REQ_REJECT	343
5.2.14 IBI_SIR_REQ_REJECT	344
5.2.15 RESET_CTRL	345
5.2.16 SLV_EVENT_STATUS	348
5.2.17 INTR_STATUS	351
5.2.18 INTR_STATUS_EN	356
5.2.19 INTR_SIGNAL_EN	359
5.2.20 INTR_FORCE	362
5.2.21 QUEUE_STATUS_LEVEL	365
5.2.22 DATA_BUFFER_STATUS_LEVEL	367
5.2.23 PRESENT_STATE	368
5.2.24 CCC_DEVICE_STATUS	372
5.2.25 DEVICE_ADDR_TABLE_POINTER	375
5.2.26 DEV_CHAR_TABLE_POINTER	377
5.2.27 VENDOR_SPECIFIC_REG_POINTER	379
5.2.28 SLV_MIPI_ID_VALUE	380
5.2.29 SLV_PID_VALUE	381
5.2.30 SLV_CHAR_CTRL	383
5.2.31 SLV_MAX_LEN	386
5.2.32 MAX_READ_TURNAROUND	387
5.2.33 MAX_DATA_SPEED	388
5.2.34 SLV_INTR_REQ	391
5.2.35 SLV_TSX_SYMBL_TIMING	394
5.2.36 DEVICE_CTRL_EXTENDED	395
5.2.37 SCL_I3C_OD_TIMING	397
5.2.38 SCL_I3C_PP_TIMING	398
5.2.39 SCL_I2C_FM_TIMING	399
5.2.40 SCL_I2C_FMP_TIMING	400
5.2.41 SCL_EXT_LCNT_TIMING	401
5.2.42 SCL_EXT_TERMN_LCNT_TIMING	403
5.2.43 SDA_HOLD_SWITCH_DLY_TIMING	405
5.2.44 BUS_FREE_AVAIL_TIMING	407
5.2.45 BUS_IDLE_TIMING	409
5.2.46 SCL_LOW_MST_EXT_TIMEOUT	410
5.2.47 I3C_VER_ID	411
5.2.48 I3C_VER_TYPE	412
5.2.49 QUEUE_SIZE_CAPABILITY	413
5.2.50 DEV_CHAR_TABLE1_LOC1	416
5.2.51 SEC_DEV_CHAR_TABLE1	417
5.2.52 DEV_CHAR_TABLE1_LOC2	418
5.2.53 DEV_CHAR_TABLE1_LOC3	419
5.2.54 DEV_CHAR_TABLE1_LOC4	420
5.2.55 DEV_ADDR_TABLE1_LOC1	421
5.2.56 DEV_ADDR_TABLE_LOC1	424
Appendix A	
Area and Power	427
A.1 Area and Power	427

Appendix B	
Synchronizer Methods	431
B.1 Synchronizers Used in DWC_mipi_i3c	432
Chapter C	
Internal Parameter Descriptions	433

Revision History

The following table provides a summary of changes made to this Databook.

Date	Version	Description
January 2020	1.00a-lca03	<p>Updated:</p> <ul style="list-style-type: none"> ■ “General Product Description” on page 18 ■ “System Block Diagram” on page 18 ■ 1.2.1 to 1.2.7 ■ “I3C System” on page 18 ■ “Master (Main Master and Secondary Master) and APB Slave” on page 23 ■ “Device Address” on page 24 ■ “Operation Modes” on page 25 ■ “Register Block Module” on page 30 ■ “Master Transaction Control Module” on page 31 ■ “Bus Monitor” on page 32 ■ “I/O Buffer Interface” on page 33 ■ “Single Port RAM Interface” on page 34 ■ “SPRAM Buffer Depth Components” on page 38 ■ “Configuring Buffer Depths” on page 38 ■ “Overview of Master Role in DWC_mipi_i3c” on page 42 ■ “SETDASA Procedure (Format 2: Point-to-point)” on page 43 ■ “ENTDAA Procedure” on page 45 ■ “In-Band Interrupt (IBI) Detection and Handling” on page 47 ■ “Transfer Command Data Structure” on page 56 ■ “Short Data Argument Data Structure” on page 59 ■ “Address Assignment Command Data Structure” on page 61 ■ “Response Data Structure” on page 62 ■ “IBI Status and Data Structure” on page 64 ■ “Device Address Table Data Structure” on page 66 ■ “I3C Device with Static Address” on page 67 ■ “I3C Device with Dynamic Address” on page 67

Date	Version	Description
January 2020	1.00a-lca03	<p>Updated:</p> <ul style="list-style-type: none"> ■ “I2C Device with Static Address” on page 67 ■ “Device Characteristics Table Data Structure” on page 68 ■ “I3C Private Write or Read Transfers” on page 95 ■ “I2C Private Write/Read Transfers Required Programming Values” on page 97 ■ “Overview of SCL Generation and Timings in DWC_mipi_i3c” on page 123 ■ “DWC_mipi_i3c Timing Registers” on page 124 ■ “SCL Extended Termination Low Count Timing Register” on page 126 ■ “Derivation of I3C/I2C Timing Parameters from Timing Registers” on page 127 ■ “Slave Role Related Registers” on page 136 ■ “CCS Handled through Configuration Parameters” on page 139 ■ “Description of Master Request (MR) Generation” on page 148 ■ “CCCs Handled through Configuration Parameters” on page 139 ■ “Short Data Argument Data Structure” on page 60 ■ Table 2-19, Table 2-20, Table 2-21, Table 2-22 ■ “Operation Modes of DWC_mipi_i3c” on page 90 ■ “Transfer Argument Data Structure” on page 59 ■ “DWC_mipi_i3c Command Structure” on page 55 ■ “SDA Hold Switch Delay Timing Register” on page 126 ■ “Overview of Slave Role in DWC_mipi_i3c” on page 135 ■ “I3C vs I2C Role Selection” on page 135 ■ “CCC Transfer Related Registers” on page 143 ■ “Slave Role Related Registers” on page 136 ■ “Overview of Private Data Transfers” on page 144 ■ “Overview of Hot-Join Request Generation with DWC_mipi_i3c Slave” on page 146 ■ “Slave Interrupt Request (SIR) Related Registers” on page 148 ■ “Values of IBI_STS” on page 148 ■ “Master Request (MR) Related Registers” on page 149 ■ “Speed and Clock requirements of DWC_mipi_i3c APB Slave” on page 26 <p>Added:</p> <ul style="list-style-type: none"> ■ “JEDEC Module Sideband Bus Configuration” on page 24 ■ “Handling of the GETSTATUS CCC” on page 142 ■ “SDA Hold Switch Delay Timing Register” on page 126 ■ “Master Data Structures in HCI Mode” on page 70 ■ “Clock Stalling Conditions” on page 117 ■ “Data Packing and Unpacking” on page 119

Date	Version	Description
January 2020	1.00a-lca03	<p>Added:</p> <ul style="list-style-type: none">■ “SETAASA Procedure” on page 46■ “CCCs Handled Through Both Ports and Registers” on page 140■ “Handling of ENTDAA, GETPID and GETDCR” on page 142■ “Defining Byte Support” on page 132■ “Packet Error Check” on page 132■ “Broadcast CCC’s in I2C Speed” on page 133■ “BUS RESET Generation” on page 133■ “Conditions for Mode Change” on page 136■ “Conditions for Glitch Filter Enable/Disable” on page 136■ “CCC Transfer Related Registers” on page 143■ “Device Address Table Registers” on page 67■ “DEV_ADDR_TABLE_LOCx” on page 68■ “DEV_ADDR_TABLEEx_LOC1” on page 68■ “Device Characteristic Table Registers” on page 70

Date	Version	Description
March 2019	1.00a-lca02	<p>Updated:</p> <ul style="list-style-type: none">■ “General Product Description” on page 18■ Figure 1-1■ “System Block Diagram” on page 18■ “Standards Compliance” on page 20■ “DWC_mipi_i3c Features” on page 21■ “DWC_mipi_i3c APB Slave Features” on page 22■ “Unsupported Protocol Features for Master and Secondary Master” on page 22■ “Unsupported Protocol Features for APB Slave” on page 23■ Table 1-1■ Table 1-2■ “Architecture of the DWC_mipi_i3c” on page 29■ “Overview of Master Role in DWC_mipi_i3c” on page 42■ “IBI Detection and Handling” on page 49■ “ENTDAA Procedure” on page 45■ Figure 2-20■ “CCC Transfer Related Registers” on page 143■ “DWC_mipi_i3c Command Structure” on page 55■ “Transfer Argument Data Structure” on page 59■ “High Data Rate-Ternary Symbol (HDR-TSP / HDR-TSL) Transfers” on page 101■ “DWC_mipi_i3c Timing Registers” on page 124■ “Bus Free and Available Timing Register” on page 126■ “SCL Extended Termination Low Count Timing Register” on page 126■ “I3C Timing When Communication with I2C Legacy Devices” on page 127■ “Handling of GETMXDS CCC” on page 141■ “Overview of Hot-Join Request Generation with DWC_mipi_i3c Slave” on page 146■ “Master Request (MR) Related Registers” on page 149■ “Data Structure in DWC_mipi_i3c Slave” on page 149■ “Response Data Structure” on page 107■ “Area and Power” on page 427

Date	Version	Description
March 2019 (Continued...)	1.00a-lca02	<ul style="list-style-type: none"> ■ Table 2-7 ■ Table 3-8 ■ Figure 2-11 ■ Figure 3-4 ■ Figure 3-5 ■ Figure 3-7 ■ Figure 3-8 ■ “Parameter Descriptions” on page 153 ■ “Signal Descriptions” on page 187 ■ “Register Descriptions” on page 213 ■ “Internal Parameter Descriptions” on page 433 ■ Appendix B.1, “Synchronizers Used in DWC_mipi_i3c” <p>Added:</p> <ul style="list-style-type: none"> ■ “Bus Communication Protocols in Non-HCI Modes” on page 90 ■ “SCL I2C Fast Mode Timing Register” on page 125 ■ “SCL I2C Fast Mode Plus Timing Register” on page 125
May 2018	1.00a-lca01	Initial release

Preface

This document describes the features and operations of Synopsys' DesignWare Cores DWC MIPI I3C Master and Slave Controller (DWC_mipi_i3c).

Databook Organization

The chapters of this databook are organized as follows:

- [Chapter 1, "Product Overview"](#) describes the overall product features of DWC_mipi_i3c.
- [Chapter 2, "Architecture of the DWC_mipi_i3c"](#) describes the architecture of DWC_mipi_i3c controller in Master and Slave modes of operation.
- [Chapter 3, "Parameter Descriptions"](#) describes parameters that allow configuration of DWC_MIPI_i3c controller.
- [Chapter 4, "Signal Descriptions"](#) describes DWC_mipi_i3c controllers I/Os.
- [Chapter 5, "Register Descriptions"](#) describes registers of DWC_mipi_i3c controller.
- [Appendix A, "Area and Power"](#) provides an estimation of DWC_mipi_i3c area values.
- [Appendix B, "Synchronizer Methods"](#) describes the synchronizer methods that are used in DWC_mipi_i3c to cross clock boundaries.
- [Appendix C, "Internal Parameter Descriptions"](#) describes internal parameters.

Reference Documentation

- MIPI® Alliance Specification I3CSM, Version 1.0
- MIPI® Alliance Specification for I3C BasicSM Version 1.0
- MIPI® Alliance Specification for I3C Host Controller Interface (I3CSM HCISM), Version 1.0 Sep-2017
- AMBA 3 APB Protocol Specification v1.0 from ARM®
- JEDEC Module Sideband Bus Specification v0.53_AA (JESD403-1), July 2019

Web Resources

- DesignWare IP product information: <http://www.designware.com>
- Your custom DesignWare IP page: <http://www.mydesignware.com>
- Documentation through SolvNet: <http://solvnet.synopsys.com> (Synopsys password required)

- Synopsys Common Licensing (SCL): <http://www.synopsys.com/keys>

Customer Support

To obtain support for your product:

- First, prepare the following debug information, if applicable:
 - For environment setup problems or failures with configuration, simulation, or synthesis that occur within coreConsultant or coreAssembler, use the following menu entry:
File > Build Debug Tar-file
Check all the boxes in the dialog box that apply to your issue. This menu entry gathers all the Synopsys product data needed to begin debugging an issue and writes it to the file <core tool startup directory>/debug.tar.gz.
 - 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.
- Then, contact Support Center, with a description of your question and supplying the above information, using one of the following methods:
 - For fastest response, use the SolvNet Web site. If you fill in your information as explained below, your issue is automatically routed to a support engineer who is experienced with your product. The Sub Product 1 entry is critical for correct routing.

Go to <http://solvnet.synopsys.com/EnterACall> and click on the link to enter a call.
Provide the requested information, including:

- Customer Tracking Number: Enter your project name. Use the same name for cases related to the same project.
- **Product:** DesignWare Cores
- **Sub Product 1:** MIPI Controller
- **Sub Product 2:** DWC MIPI I3C
- **Product Version:** Version 1.00a-lca03
- **Problem Type:**
- **Issue Severity:**
- **Problem Title:** Provide a short summary of the issue or list the error message you have encountered
- **Problem Description:** For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood

After creating the case, attach any debug files you created in the previous step.

- Or, send an e-mail message to support_center@synopsys.com (your e-mail will be queued and then, on a first-come, first-served basis, manually routed to the correct support engineer):
 - Include the Product name, Sub Product 1 name, Sub Product 2 name, and Product Version number in your e-mail (as identified above) 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 in the previous step.

- 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:
<http://www.synopsys.com/Support/GlobalSupportCenters>

Product Overview

This chapter introduces the Synopsys DesignWare Cores DWC MIPI I3C Master and Slave Controller (DWC_mipi_i3c) features and deliverables. It contains the following sections:

- “General Product Description” on page [18](#)
- “DWC_mipi_i3c Features” on page [21](#)
- “Speed and Clock Requirements” on page [26](#)
- “Deliverables” on page [26](#)

1.1 General Product Description

The Improved Inter Integrated Circuit (I3C) is part of a group of communication protocols defined by the MIPI® Alliance. This specification is developed to ease sensor system design architectures in mobile wireless products by providing a fast, low cost, low power, two-wire digital interface for sensors.

The DesignWare Cores DWC MIPI I3C Master and Slave Controller (referred to as DWC_mipi_i3c) - Dual Role Device - is a digital controller that implements all protocol functions related to Master (Main Master and Secondary Master) and Slave as defined in the MIPI I3C Specification. The DWC_mipi_i3c provides an interface between the system (application) and the I3C Interface, allowing the communication with the I3C devices and Legacy I2C devices (slaves), with limitations as specified in MIPI I3C Specification.

In Master mode of operation, for a specific configuration of DWC_mipi_i3c, a HCI Interface between the system (application) and DWC_mipi_i3c, as specified in MIPI I3C HCI 1.0 Specification is available.

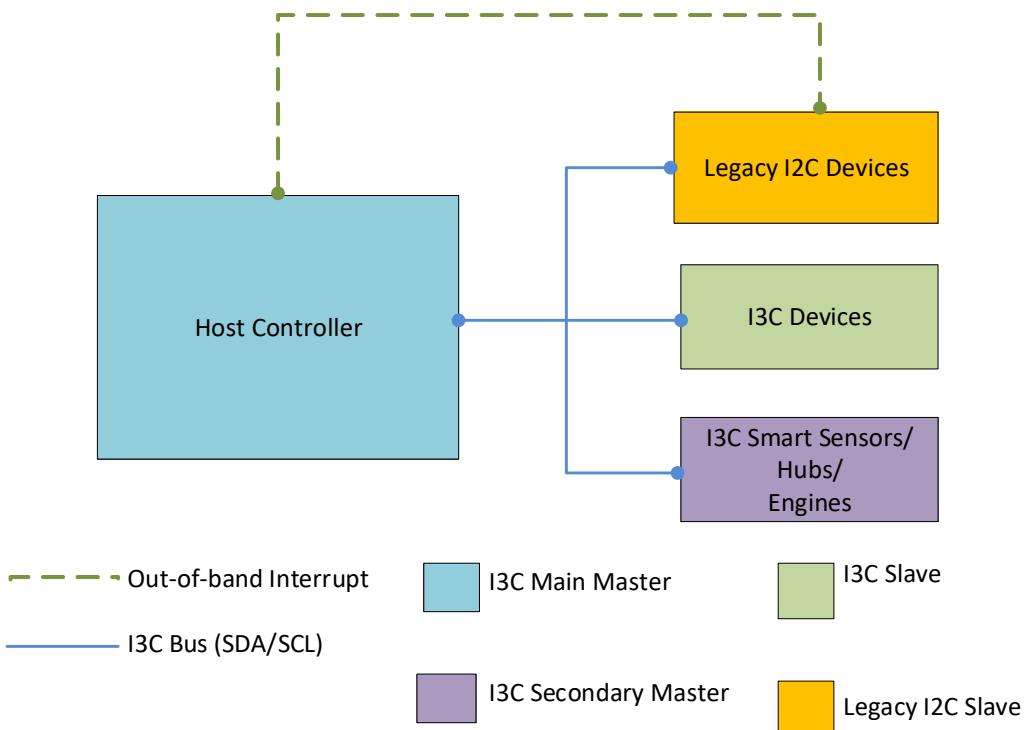
The Dual Role DWC_mipi_i3c controller can be configured as I3C Main/Secondary Master or I3C Slave with the processor interface (APB). The controller can also be configured as I3C Slave Lite with processor-less interface. The controller meets the requirement of I3C protocol and I2C protocol as specified by the MIPI I3C Specification.

1.1.1 System Block Diagram

The I3C system ensures reliable transmission and reception of data. The I3C interface is intended to improve upon the features of the I2C interface, while preserving backward compatibility, with some limitations (for example, clock stretching, clock synchronization, high-speed mode, and 10-bit addressing is not supported).

[Figure 1-1](#) shows the block diagram of I3C System.

Figure 1-1 I3C System



The I3C Protocol supports the following different roles of devices:

- Main Master
- Secondary Masters
- I3C Slaves
- I2C Slaves

The Main Master is a specialized Master that comes up after power-on-reset, and is responsible for assigning dynamic addresses to the I3C devices. The Secondary Master is an I3C instance capable of both Master and Slave functionality. It comes up as a slave upon power-on-reset. The Secondary Master must get the ownership of the I3C bus to become a current Master before initiating any transfer to its associated Slaves.

1.1.2 Applications

Typical applications built with DWC_mipi_i3c are mobile SoC, application processors, and co-processors, that target the following applications:

- Smart phone
- Camera
- Touch screen
- Wearables
- Memory Interfaced applications
- Automotive, and so on

The following are some of the typical sensor classes addressed by I3C, and therefore can be built with DWC_mipi_i3c:

- Motion Sensors
 - Gyro
 - Touch screen
 - Grip
 - Time of Flight (sensors)
- Biometric Sensors
 - Fingerprint
 - Heart Rate
- Environmental Sensors
 - Ambient Light
 - Temperature
- NFC (Near Field Communication)

1.1.3 Standards Compliance

The dual role DWC_mipi_i3c conforms to the following standards:

- MIPI® Alliance Specification for Improved Inter Integrated Circuit (I3CSM), Version 1.0 Dec-2016
- MIPI® Alliance Specification for I3C BasicSM Version 1.0
- AMBA 3 APB Protocol Specification v1.0 from ARM®
- MIPI® Alliance Specification for I3C Host Controller Interface (I3CSM HCISM), Version 1.0 Sep-2017
- JEDEC Module Sideband Bus Specification v0.53_AA (JESD403-1), July 2019

1.2 DWC_mipi_i3c Features

1.2.1 General Features

The Dual Role DWC_mipi_i3c supports the following general features:

- Two-Wire I3C serial interface - consists of a Serial Data Line (SDA) and a Serial Clock (SCL)
- Configurable Device Roles:
 - I3C Master (Single Master)
 - I3C Secondary Master
 - I3C Slave
 - I2C Slave
- Separate command and data buffers for ease of transfers
- Configurable Command, Response and Data Buffer Depths
- Single port RAM support for Command and Data Buffers
- Supports up to 2^{16} (65536) Write or Read bytes with a single command
- Hardware assisted Dynamic Address Assignment (DAA) support
- Hardware assisted device role switching in Secondary Master configuration
- Hot-Join support with user controllable filter
- Supports Data transfer to legacy I2C slaves
- Supports various Data rates (FM, FM+, SDR, HDR-DDR, TSP/TSL)
- CRC/Parity Generation and Validation
- Support for broadcast and directed CCC transfers
- APB3 Slave Interface for register access
- External DMA support through hardware handshake interface
- Debug Interface

1.2.2 DWC_mipi_i3c Master Features

The following features are specific to Dual Role DWC_mipi_i3c configured as I3C Master and Secondary Master:

- Autonomous clock stalling support in Master mode
- Supports Device Address Table for addressing multiple Slaves.
- Dedicated buffer for capturing information from ENTDAA CCC in current Master mode and information from DEFSLVS CCC in non current master mode.
- Detects arbitration loss due to incoming IBI and subsequently re-transmits the command.
- Use of Duty Cycle to achieve Lower Effective Speed for SDR transfers to work with slower I3C slaves.

- Programmable SDA Transmit Hold
- Programmable Retry Count for transfers that are NACK'ed (Address) by Slaves
- In-Band Interrupt with MDB/Payload support in I3C Master (Single Master) Mode only
- In-Band Interrupt without MDB/Payload support in Secondary Master Mode
- Defining Byte Support for vendor specific Broadcast and Directed CCC Transfers.
- JESD403-1 (JEDEC Sideband specification) additional features in single Master mode (IC_DEVICE_ROLE==1)
 - Address Assignment through SETAASA CCC Command
 - Generation of Broadcast CCC Transfers in I2C Speed
 - SCL Timed reset and HDR Exit pattern based reset generation
 - Generation and Validation of PEC Byte for
 - Vendor Specific Broadcast and Directed CCC Transfers
 - Private Write and Read SDR Transfers
 - IBI With Payload Transfers
- Compliance to mandatory features of HCI 1.0 in Single Master working in PIO mode only.

1.2.3 DWC_mipi_i3c APB Slave Features

The following features are specific to DWC_mipi_i3c configured as I3C APB Slave:

- Static or Dynamic Slave Device Support
- Built-in Hardware Dynamic Address Allocation support (ENTDAA/SETDASA)
- Built-in CCC transfer handler (does not involve the application)
- Configurable Queue Depths
- Configurable Data Buffer Depths
- Auto Hot-Join request generation support
- Slave Interrupt Request generation support
- Master Request generation support
- I2C mode support
- Adaptive mode of operation between I2C and I3C depending on I3C bus traffic
- Built-in S0-S5 Error Handling

1.2.4 DWC_mipi_i3c Slave-Lite Features

Refer "Slave-Lite Features" section in *DWC_mipi_i3c Slave Lite databook*.

1.2.5 Unsupported Protocol Features for Master and Secondary Master

- Optional Address Arbitration Optimization

- Optional Timing Control
 - Synchronous Systems and Events
 - Asynchronous Systems and Events
- Bridge Device Capability
- In-Band Interrupt (with MDB/Payload) support in Secondary master mode.
- HCI Features
 - DMA support in HCI mode of operation.
 - Auto Command at HDR speeds HCI mode of operation.
 - Combo transfers at HDR and mixed speeds (For example, DDR-DDR, TS-TS, SDR-DDR, SDR-TS, and so on) HCI mode of operation.
 - Timestamping support HCI mode of operation.

1.2.6 Unsupported Protocol Features for APB Slave

- Slave interrupt request support with data
- Sub-address/Register-address decoding
- Vendor specific CCC
- Random Value for Provisional ID Type Selector

1.2.7 Interfaces

The Dual role DWC_mipi_i3c has the following interfaces:

- AMBA APB Slave Interface
- I3C SDA/SCL IO buffer interface for external IO buffer connections
- Single-Port RAM (SPRAM) interface for an external RAM connection
- DMA handshaking interface compatible with the DW_ahb_dmac handshaking interface
- External CDR interface for TSP/TSL mode of operation

1.2.8 Master (Main Master and Secondary Master) and APB Slave

The Master in I3C system supports Dynamic Address Assignment (DAA). The Master is responsible for generating the clock, issuing the commands, and controlling the data transfer.

Each I3C device has a unique address that is assigned by the Master through Dynamic Address Assignment (DAA) during cold power-up or during a Hot-Join of I3C device to the system.

The Main Master is the current Master which has ownership of the bus during cold power-up.

The Slave is responsible for responding to the commands, transmitting or receiving data to/from the Master, and acknowledging to the Master that the transfer is successful.

1.2.9 Bus Configuration

DWC_mipi_i3c supports three types of I3C bus configuration, as defined by the MIPI I3C Specification:

- Pure Bus – The configuration with only I3C devices present on the bus.
- Mixed Fast Bus – The configuration with both I3C devices and Legacy I2C Slaves present on the bus. In this case, the legacy I2C Slaves are restricted to the ones that are generally permissible (that is, Slave-only, no clock stretching and that have a true I2C 50ns glitch filter on SCL).
- Mixed Slow/Limited Bus – The configuration with both I3C devices and legacy I2C Slaves present on the bus. In this case, the legacy I2C Slaves are restricted to the ones that are selectively backward compatible with the I2C standard (that is, Slave-only and no clock stretching); but, these do not have a true I2C 50 ns glitch filter on SCL.

In a Mixed bus configuration, the maximum possible data rate with I3C devices depends on the compliance of the I2C Slave as defined by the I2C Specification. The maximum data rate as specified in I3C specification is possible only if all I2C slaves have the 50 ns spike filter.

In the absence of spike filters or if the presence of filter is unknown, the maximum data rate is limited to only FM or FM+, even for I3C devices (as per the I3C Specification). I2C Slaves are not allowed to extend the clock.

1.2.9.1 JEDEC Module Sideband Bus Configuration

JEDEC Module Sideband specification (JESD403-1) is based on MIPI I3C Basic Specification. The DWC_mipi_i3c supports JEDEC Sideband specification features in the Master-Only configuration. The features are compliant to the JESD403-1 draft specification as mentioned in the section “Reference Documentation” of the databook. To support the Sideband specification, select the following JEDEC Sideband Specific parameters during configuration:

- Parameter IC_HAS_DEVICE_RESET_SUPPORT: Timed RESET can be enabled through this configuration parameter. Once the controller is configured for "Timed Reset" support, the usage can be controlled through programming.
- Parameter IC_HAS_PEC: The Packet Error Check (PEC) Generation/Validation function support can be enabled through configuration parameter. Once the controller is configured for the PEC support, the usage can be controlled through programming.

You can issue the following transfer commands from the Master application to communicate with JEDEC Sideband compliant slave devices.

- Address Assignment through SETAASA CCC
- Generation of JEDEC Sideband specific CCCs like DEVCAP, SETHID, DEVCTRL

Support for JESD403-1 features in DWC_mipi_i3c are limited to one configuration that is representative of the JEDEC use case and can be found under “Validated Configurations” section of *DWC_mipi_i3c Controller User Guide*.

1.2.10 Device Address

In I3C system, the static address devices are the ones which are enhanced from I2C system to I3C, and they also obtain dynamic address.

The dynamic address devices are the ones which are developed for I3C system, and they do not have any I2C static address.

For the Main Master, the dynamic address is always assigned before enabling the controller (self-assigned Dynamic Address). All other devices (Slaves and secondary masters) obtain the dynamic address from the Main Master after they are enabled through any of the address assignment CCC such as ENTDAA, SETDASA, or SETAASA.

1.2.11 In-Band Interrupt (IBI)

The I3C protocol also gives provision of supporting In-Band interrupts within the two-wire interface, which reduces the device pin count and signal paths. The In-Band interrupt is initiated by the I3C Slave device to the current Master.

Additionally, I3C uses In-Band interrupt for generating Hot-Join request (for requesting the host to assign I3C Dynamic Address) and mastership request to request bus ownership.

1.2.12 Operation Modes

The DWC_mipi_i3c supports the following device roles that are configurable at the time of generating RTL:

- Device Roles
 - I3C Master: Enables the DWC_mipi_i3c to support Master role (Single Master).
 - I3C Slave: Enables the DWC_mipi_i3c to support Slave role (Slave-Lite or APB Slave)
 - Secondary Master: Role-agnostic instance that can change the role from Master to Slave and vice versa through protocol semantics.
- In this configuration, the role of the instance can be assigned by software programming also. This is referred to as the Programmable Master Slave.
- Notes on Device Roles
 - Single Master: Indicates that DWC_mipi_i3c is the only Master in the I3C system.
 - I3C APB Slave: Indicates that DWC_mipi_i3c is an APB Slave in the I3C system
 - Slave Lite: Enables the DWC_mipi_i3c to support a non-processor Interface. Also available as a separate product. Contact Synopsys for details about the I3C Slave-Lite product.

1.3 Speed and Clock Requirements

1.3.1 Speed and Clock requirements of DWC_mipi_i3c Master

[Table 1-1](#) shows the clock frequencies of DWC_mipi_i3c Master.

Table 1-1 DWC_mipi_i3c Master Clock, Programmable Master Slave, and Secondary Master Domains and Frequencies

Clock Domain	Minimum Frequency	Typical Frequency	Maximum Frequency
core_clk (Core Clock)	125 MHz	125 MHz	700 MHz
pclk (Application Clock)	30 MHz	50 MHz	700 MHz
dma_clk (DMA Clock)	30 MHz	50 MHz	700 MHz

For details on integrating the clock domains, refer to *DWC MIPI I3C Controller User Guide*.

1.3.2 Speed and Clock requirements of DWC_mipi_i3c APB Slave

[Table 1-2](#) shows the clock frequencies of DWC_mipi_i3c Slave. Note that 'core_clk' absent in a Slave-only (IC_DEVICE_ROLE==4) configuration.

Table 1-2 DWC_mipi_i3c Slave Clock Domains and Frequencies

Clock Domain	Minimum Frequency	Typical Frequency	Maximum Frequency
pclk (Application Clock)	30 MHZ	100 MHZ	700 MHz
hdr_tx_clk (HDR Transmit Clock)	25 MHZ	100 MHZ	700 MHz



Note For TSP/TSL mode of operation, a clock data recovery circuit should provide the HDR clk for the HDR TSP/TSL receive logic.

For details on integrating the clock domains, refer to *DWC MIPI I3C Controller User Guide*.



Note The minimum frequency of 'pclk' and 'hdr_tx_clk' supported is 100 MHz, if synchronizer depth selected is 3 (IC_SYNC_DEPTH=3) for all Master, Slave and Secondary Master configuration.

1.4 Deliverables

The DWC_mipi_i3c requires the Synopsys coreConsultant tool to install, unpack, and configure the controller. The license file required to install, unpack, and configure DWC_mipi_i3c is delivered separately.

The DWC_mipi_i3c image includes the following:

- Verilog RTL source code
- Synthesis scripts for Synopsys Design Compiler
- Verification testbench and test configurations derived from the parameter choices made during controller configuration
- DWC_mipi_i3c ASIC and FPGA synthesis, design-for-test, timing analysis, formal verification, and timing information

2

Architecture of the DWC_mipi_i3c

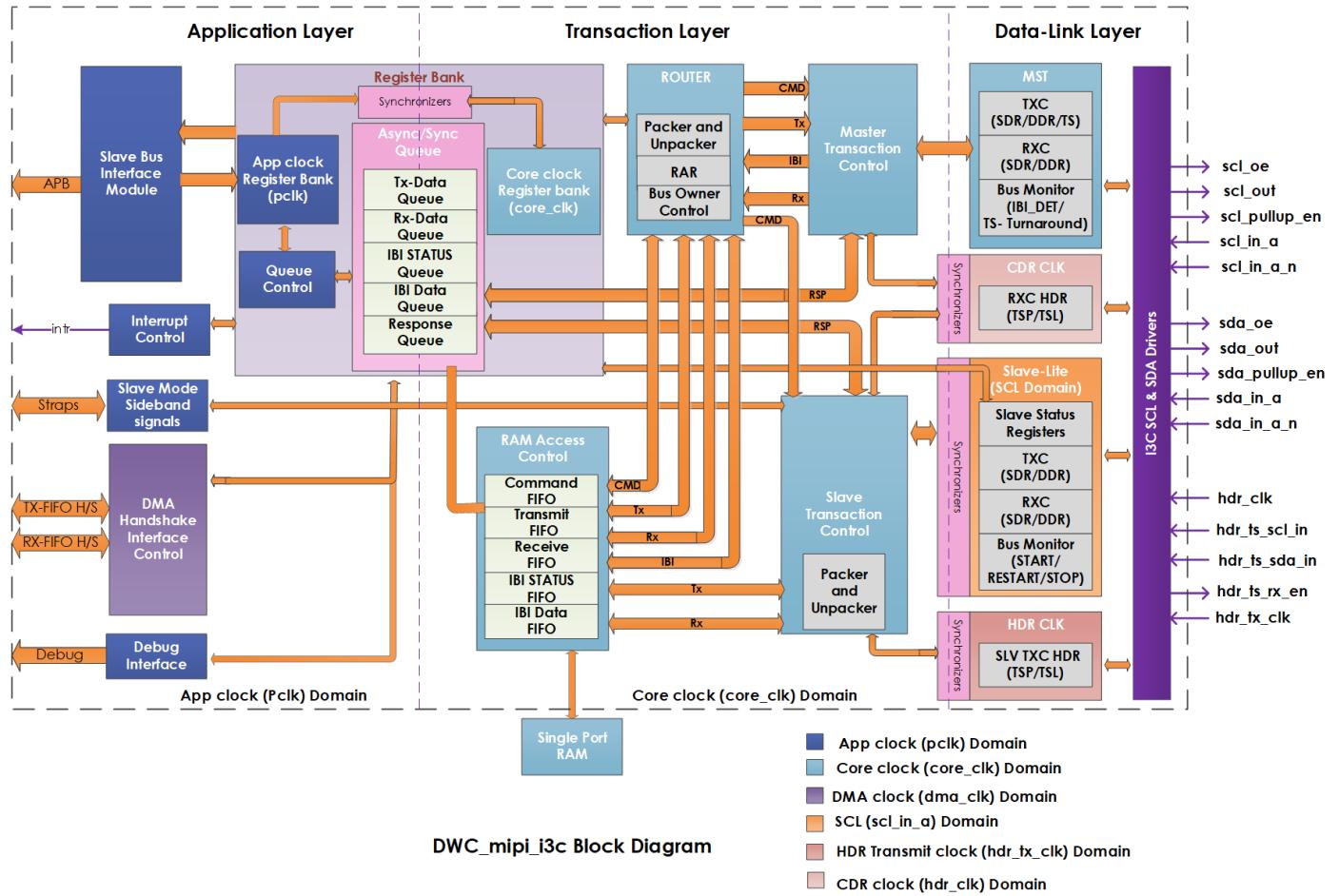
This chapter describes the DWC_mipi_i3c architecture. It contains the following sections:

- “[Block Diagram and Description of DWC_mipi_i3c](#)” on page [30](#)
- “[Description of DWC_mipi_i3c Bus Interfaces](#)” on page [33](#)
- “[Simple-Transfer DMA \(SDMA\) Handshake](#)” on page [35](#)
- “[SPRAM and Buffer Depth in DWC_mipi_i3c](#)” on page [38](#)
- “[Interrupts in DWC_mipi_i3c](#)” on page [40](#)
- “[Master Functionality with DWC_mipi_i3c](#)” on page [42](#)
- “[Slave Functionality with DWC_mipi_i3c](#)” on page [135](#)

2.1 Block Diagram and Description of DWC_mipi_i3c

Figure 2-1 shows the block diagram of DWC_mipi_i3c.

Figure 2-1 Block Diagram of DWC_mipi_i3c



2.1.1 Slave Bus Interface Module

The Slave interface module converts the APB3 transactions to the generic register interface to interact with the register block.

2.1.2 Register Block Module

The Register block contains all the DWC_mipi_i3c registers. All registers in DWC_mipi_i3c are 32-bit wide. The register interface in DWC_mipi_i3c is used to program the controller for different bus modes, timing parameters and enabling/disabling of interrupts.

The register interface is also used for the transmission of command and Tx-data to the controller and reception of Rx-data, IBI-data and response from the controller.

The Register block supports port registers, which are mapped to either FIFO's or Queue's present in the DWC_mipi_i3c controller.

2.1.3 Interrupt Control Module

The Interrupt control module enables the interrupts as a single pin. Interrupts are generated and cleared based on the Interrupt Register settings.

2.1.4 DMA Handshake Interface Module

The DMA handshake interface module is responsible for the assertion and de-assertion of the DMA handshake interface, based on the occupancy levels of Tx-FIFO and Rx-FIFO's.

2.1.5 Async Queue Block

The Async queue block consists of each asynchronous queue for the Data Buffers and Queues to maintain coherency between the Slave interface clock and core clock.

2.1.6 RAM Access Control

The RAM access control block performs arbitration role for the data to be communicated between:

- Data Router and SPRAM
- APB interface and SPRAM

2.1.7 Data Router Block

Data router block performs the packing and unpacking of data between transaction layer and the RAM access control block.

Data that comes as separate bytes is packaged and written to the RAM access control that supports 4-byte data. Similarly, data transmitted from RAM access control (4-byte data) is also unpacked to separate bytes or words, as required by the transaction layer.

2.1.8 Master Transaction Control Module

The Master transaction control block is a state machine that controls both transmit and receive operations in the current Master mode of DWC_mipi_i3c. This module instructs the transmit and receive control blocks to operate based on commands from the application interface or In-band interrupt from the slaves.

2.1.9 Slave Transaction Control Module

The Slave transaction control block is a state machine that controls both transmit and receive in the Slave and non-current master mode of DWC_mipi_i3c. This module instructs the transmit and receive control blocks based on incoming transfer from the Master or the command from the application to drive the In-Band Interrupts.

2.1.10 Transmit Control and Receive Control Blocks

The Transmit control and receive control blocks are used to control protocol-related transmit or receive signals (including timing signals) for Master and Slave, and also for arbitration loss checks.

2.1.11 Bus Monitor

The Bus Monitor block is divided into two modules; one for Master and another for Slave mode.

- IBI Start and HDR-TS Turnaround detection in Master Mode.
- START, RESTART and STOP conditions in Slave Mode.
- Bus-free (in Master mode), Bus-available (in Slave mode) and Bus-Idle (in Slave mode) conditions.

2.2 Description of DWC_mipi_i3c Bus Interfaces

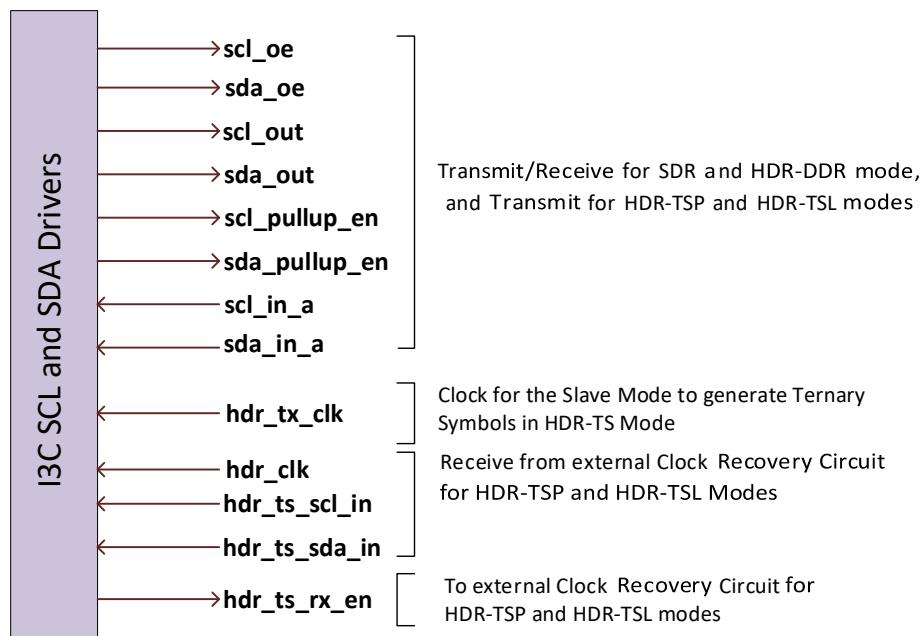
The DWC_mipi_i3c controller consists of the following interfaces:

- I/O buffer interface
- HDR clock recovery interface
- APB interface
- Interrupt interface
- DMA handshake interface for external DMA
- Single port RAM interface
- Debug interface

2.2.1 I/O Buffer Interface

The signals for connecting external drivers and I/O buffers to the SDA line, SCL line, and the various clocks, can be grouped as shown in [Figure 2-2](#).

Figure 2-2 Grouping of Signals



The signal descriptions are as follows:

- scl_oe and sda_oe signals are used to enable the SCL and SDA Pads respectively.
- scl_out and sda_out signals are clock and data output signals of the DWC_mipi_i3c interface from the DWC_mipi_i3c controller.
- scl_pullup_en and sda_pullup_en signals are used to enable the Pull-Up resistor or equivalent current source in Open-Drain mode when the controller requires to drive 1.

- scl_in_a and sda_in_a signals are the clock and data input signals of the I3C interface to the DWC_mipi_i3c controller.
- hdr_clk is the recovered clock signal generated from the external clock recovery circuit (CDR) block.
- hdr_tx_clk is transmit clock used by the slave to generate the Ternary Symbols in HDR-TSP and HDR-TSL Modes.
- hdr_ts_scl_in and hdr_ts_sda_in are SCL and SDA signals synchronized to the recovered clock (hdr_clk) in HDR-TSP and HDR-TSL Modes.
- The hdr_ts_rx_en signal enables the external Clock Recovery Circuit (CDR) to generate the recovered clock based on incoming SCL and SDA signal changes.



Note The SCL clock signal is always in push-pull mode and SDA data signal can switch between push-pull and open-drain mode.

2.2.2 APB Interface

DWC_mipi_i3c supports APB3 application slave interfaces.

2.2.3 Interrupt Interface

DWC_mipi_i3c generates a single interrupt which consolidates (logical OR'd) all interrupts in Interrupt Status register.

2.2.4 DMA Handshake Interface

DWC_mipi_i3c consists of DMA handshake interface for connecting to external DMA, that is, DW_ahb_dmac or DW_axi_dmac. The DMA handshake interface is provided only for Tx and Rx FIFOs.

2.2.5 Debug Interface

The Debug signals are available to assess the internal states of DWC_mipi_i3c.

2.2.6 Single Port RAM Interface

The Single Port RAM (SPRAM) in DWC_mipi_i3c is a shared memory space for storing the following:

- Device address table
- Device characteristics table
- Command FIFO
- Tx and Rx FIFOs
- IBI Status and Data FIFO's

The size of SPRAM depends on the configuration parameters related to the tables in the list.

2.3 Simple-Transfer DMA (SDMA) Handshake

2.3.1 Overview of SDMA Handshake in DWC_mipi_i3c

DWC_mipi_i3c has an optional signal interface to control External DMA Controller. This feature can be selected at configuration time.

It has a handshaking interface to an external DMA Controller to request and control transfers. These transfers are exclusively for fetching of the transmit and the receive data and not for fetching the I3C command or for posting of the I3C response to the memory.

2.3.2 Description of SDMA Handshake in DWC_mipi_i3c

The APB bus is used to perform the data transfer to or from the DMA. While the DWC_mipi_i3c external DMA control operations are designed in a generic way to fit any DMA controller as easily as possible, it is designed to work seamlessly, and best used, with the DesignWare DMA controller, the DW_ahb_dmac.

The settings of the DW_ahb_dmac that are relevant to the operation of the DWC_mipi_i3c are explained in this section, mainly bit fields in the DW_ahb_dmac channel control register, CTLx, where x is the channel number.



Note When DWC_mipi_i3c operates in a master mode communicates with an external DW_ahb_dmac, the DW_ahb_dmac is always the flow controller for the TX and RX channels, that is, it controls the block size. The block size must be programmed by the software in DW_ahb_dmac. DW_ahb_dmac always transfers data using DMA burst transactions if possible, for efficiency. For more information, see the DesignWare DW_ahb_dmac Databook. When DWC_mipi_i3c operates in a slave mode, the external DW_ahb_dmac is the flow controller for the TX channel and DWC_mipi_i3c is the flow controller for the RX channel.

2.3.2.1 Overview of Operation

As a block flow control device, the external DMA controller is programmed by the processor with the number of data items (block size) that are to be transmitted or received by the DWC_mipi_i3c. This is programmed in the BLOCK_TS field of the DW_ahb_dmac CTLx register.

The block is broken into a number of transactions, each initiated by a request from DWC_mipi_i3c.

The DMA Controller must also be programmed with the number of data items to be transferred for each DMA request. This is also known as burst transaction length and is programmed into the SRC_MSIZE/DEST_MSIZE fields of the DW_ahb_dmac CTLx register.

The programmed values for the MSIZE must be same as that programmed in DATA_BUFFER_THLD_CTRL register in RX_BUF_THLD and TX_EMPTY_BUF_THLD fields for source and destination respectively.

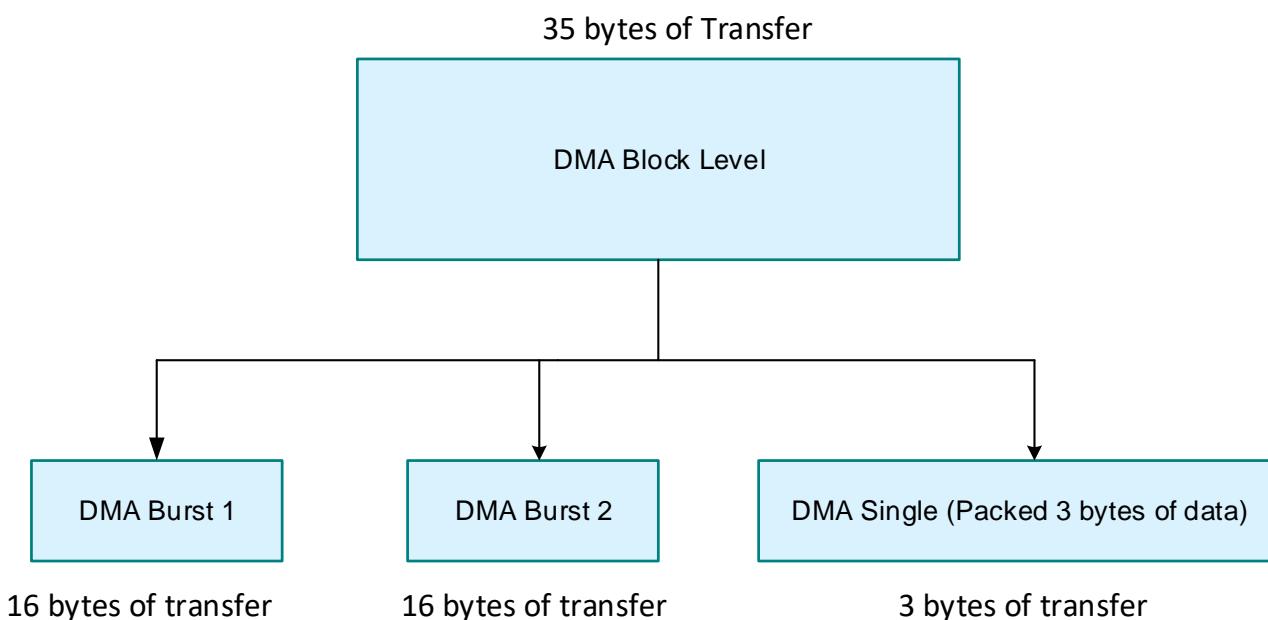
The following definitions pertain to programming the DW_ahb_dmac with respect to handshaking with DWC_mipi_i3c:

- Source single transaction size in bytes, $\text{src_single_size_bytes} = \text{CTLx.SRC_TR_WIDTH}/8$.
- Source burst transaction size in bytes, $\text{src_burst_size_bytes} = \text{CTLx.SRC_MSIZE} * \text{src_single_size_bytes}$.

- Destination single transaction size in bytes, $\text{dst_single_size_bytes} = \text{CTLx.DST_TR_WIDTH}/8$.
(In case of DWC_mipi_i3c, the TR_WIDTH is 32, as 32-bit APB bus is being used. Hence, $\text{dst_single_size_bytes} = 4$. In other words, for every single request to DW_ahb_dmac, 4 bytes of data is transferred)
- Destination burst transaction size in bytes $\text{dst_burst_size_bytes} = \text{CTLx.DEST_MSIZE} * \text{dst_single_size_bytes}$.
(For DWC_mipi_i3c, $\text{dst_burst_size_bytes} = \text{CTLx.DEST_MSIZE}$ (equal to Tx_EMPTY_BUF_THLD or RX_BUF_THLD) * 4)
- Block size in bytes: With the DW_ahb_dmac as the flow controller, the processor programs the DW_ahb_dmac with the number of data items (block size) of source transfer width (CTLx.SRC_TR_WIDTH) to be transferred by the DW_ahb_dmac in a block transfer; this is programmed into the CTLx.BLOCK_TS field. Therefore, the total number of bytes to be transferred in a block is: $\text{blk_size_bytes_dma} = \text{CTLx.BLOCK_TS} * \text{src_single_size_bytes}$.

Figure 2-3 shows a single block transfer, where the data to be transferred by the DW_ahb_dmac is 35 bytes. The source is the memory and the destination peripheral is DWC_mipi_i3c with the DW_ahb_dmac as the flow controller.

Figure 2-3 Single Block Transfer



Assuming $\text{src_single_size_bytes} = 4$, $\text{CTLx.BLOCK_TS} = 9$.

Tx_EMPTY_BUF_THLD is programmed to four locations; hence, CTLx.DEST_MSIZE is programmed to 4. Refer to Table 6-2 of DesignWare DW_ahb_dmac Databook for MSIZE decoding.

When the block size programmed into the DMA Controller is not a multiple of the burst transaction length, a series of burst transactions followed by single transactions are needed to complete the block transfer.

Burst Transfer

For every request from the DWC_mipi_i3c, the DW_ahb_dmac is going to provide with dst_burst_size_bytes of data, which is 16 bytes. Two separate burst transfer request must be provided from the DWC_mipi_i3c to get two bursts of 16 bytes of data.

The DW_ahb_dmac ignores the dma_single_tx request if asserted from DWC_mipi_i3c during the burst transfer period. The DWC_mipi_i3c generates the burst request to the DMA (asserting dma_req_tx) whenever the number of available locations in the Transmit FIFO is greater than or equal to Tx_EMPTY_BUF_THLD value.

Single Transfer

The DW_apb_dmac, after transferring the 32 bytes in burst, enters the single transfer region. It samples the dma_single_tx, and if it is asserted, transfers the remaining three bytes of data in single transfers which is packed into 4 bytes (the MSB being invalid) of data to the input to the DWC_mipi_i3c as the DST_TR_WIDTH is 32 as mentioned in Burst Transfer for DWC_mipi_i3c.

The DWC_mipi_i3c asserts the dma_single_tx when there is at least one free entry in the transmit FIFO and clear when the transmit FIFO is full.

2.3.3 Configuring SDMA in DWC_mipi_i3c

To configure SDMA in DWC_mipi_i3c, use the following parameters during the Specify Configuration Activity in coreConsultant:

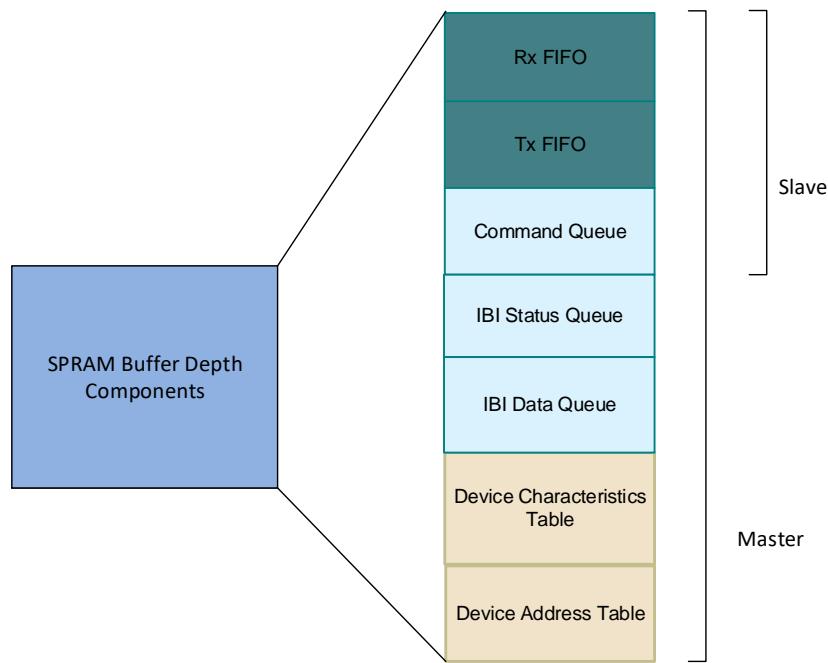
- *Include DMA handshaking interface signals* – Enable this such that the SDMA handshaking control signals to the external DMA are present in the I/Os.
- *DMA clock to Core clock* - Select a value (either 0 or 1 – Identical or Asynchronous) to define the relationship between DMA clock and core clock. If the clocks are identical (0), then the synchronization structure is removed, else the synchronization structure is in place.
- *Core clock to DMA clock Synchronization depth* – Enter a value for this parameter to define the synchronization depth of the synchronizers.
- *DMA clock to Application clock* - Select a value (either 0 or 1 - Identical or Asynchronous) to define the relationship between DMA clock and Application clock. If the clocks are identical (0), then the synchronization structure in the design is optimized. If the clocks are identical, the synchronization structures is included where ever necessary.
- *Application clock to DMA clock Synchronization depth* - Enter a value for this parameter to define the synchronization depth of the synchronizers.

2.4 SPRAM and Buffer Depth in DWC_mipi_i3c

2.4.1 Overview of SPRAM and Buffer Depth in DWC_mipi_i3c

The size of Single Port RAM (SPRAM) in DWC_mipi_i3c depends on the buffer depths assigned for different components, as shown in [Figure 2-4](#).

Figure 2-4 SPRAM Buffer Depth Components



2.4.2 Configuring Buffer Depths

The total depth of SPRAM is calculated as follows:

$$\begin{aligned} \text{IC_RAM_DEPTH} = & \text{IC_CMD_BUF_DEPTH} + \text{IC_TX_BUF_DEPTH} + \text{IC_RX_BUF_DEPTH} + \\ & \text{IC_IBI_STS_BUF_DEPTH} + \text{IC_IBI_DATA_BUF_DEPTH} + \text{IC_DEV_ADDR_TABLE_BUF_DEPTH} + \\ & \text{IC_DEV_CHAR_TABLE_BUF_DEPTH} \end{aligned}$$

The values of these buffer depths can be set during the *Specify Configuration* activity in coreConsultant, using the following parameters:

- *Command Queue Depth*
- *Transmit Data Buffer Depth*
- *Receive Data Buffer Depth*
- *Device Address Table Depth*
- *Device Characteristics Table Depth*
- *IBI Status Buffer Depth*
- *IBI Data Buffer Depth*



In Slave-Only Configuration, the Device Address Table Depth, IBI Status Buffer Depth, IBI Data Buffer Depth, and Device Characteristic Table Depth are not applicable and they do not contribute to the IC_RAM_DEPTH Parameter.

In Master Configuration, the IBI Status Buffer Depth and IBI Data Buffer Depth are not applicable if IBI With Data parameter is not selected and they do not contribute to IC_RAM_DEPTH parameter.

2.5 Interrupts in DWC_mipi_i3c

2.5.1 Overview of Interrupts in DWC_mipi_i3c

DWC_mipi_i3c supports combined interrupt, and it is synchronous to the Application Slave Interface.

2.5.2 Description of Interrupt Mechanism in DWC_mipi_i3c

The INTR_STATUS register is associated with error and status condition reporting. This register triggers the interrupt signal which is synchronous with the Slave Interface clock.

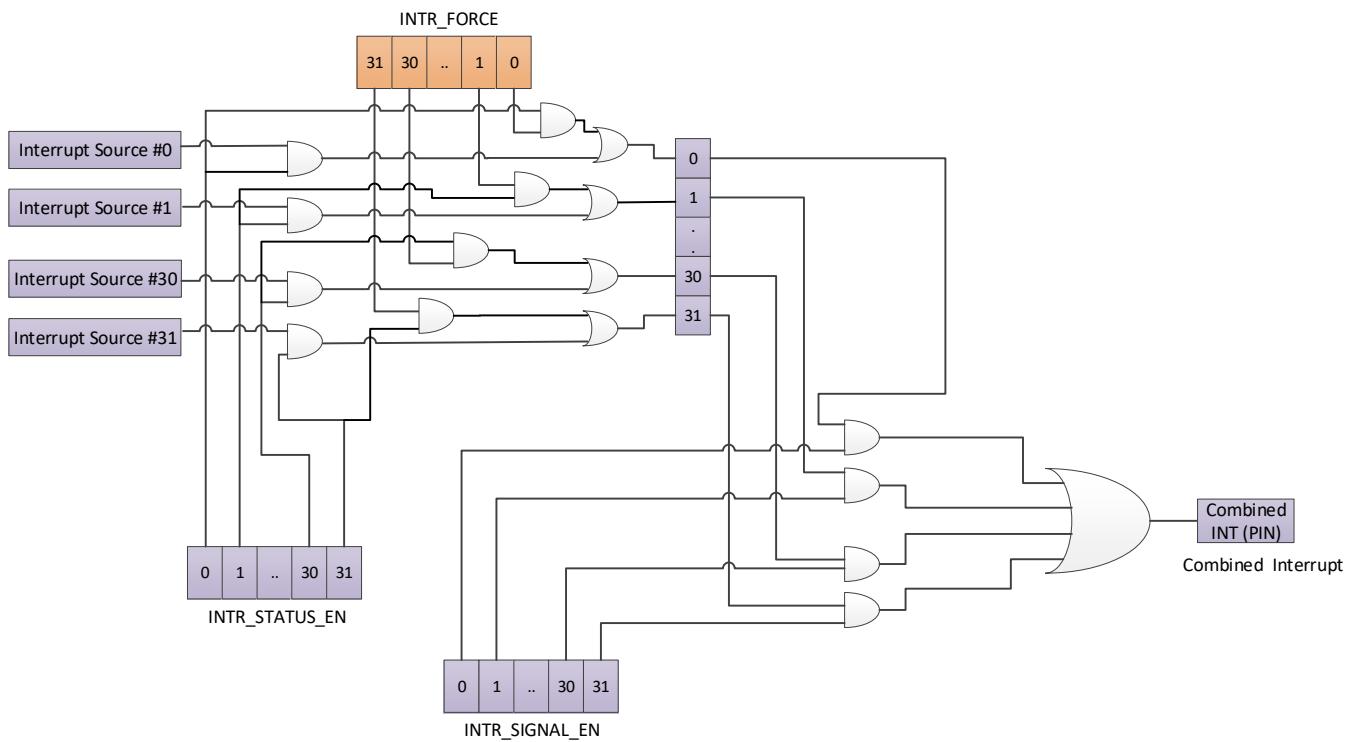
The triggering of the interrupt pin can be disabled by the INTR_SIGNAL_EN register. By default, all interrupts are enabled. When any bit of this register is set to 0, it disables the generation of interrupt in that specific interrupt pin. Thus, the interrupt signal outputs can be controlled by INTR_SIGNAL_EN register.

The interrupt bit is always set in the INTR_STATUS register irrespective of INTR_SIGNAL_EN register. The status interrupts (INTR_STATUS[4:0]) are always auto-cleared. The remaining event based interrupts (INTR_STATUS[10:5]) are cleared through writing 1'b1 to specific bit in the INTR_STATUS register.

The Interrupt Force register (INTR_FORCE) is used for test purposes, and allows triggering interrupt events individually, without the need to activate the conditions that trigger the interrupt sources, since it may be extremely complex to generate stimuli for that purpose. This feature also facilitates the development and testing of the software associated with the interrupt events. Setting any bit of these registers to 1 triggers the corresponding interrupt, provided the corresponding bit in the INTR_STATUS_EN register is set.

The Interrupt Status register (INTR_STATUS) is used for checking the true status of the interrupt events which are activated through specific bits in the INTR_STATUS_EN register. This register consists of the status of interrupt sources irrespective of disabling in INT_SIGNAL_EN register. When any bit of INTR_STATUS_EN is set to 0, it disables the generation of interrupt in that specific INTR_STATUS register bit and hardware interrupt pin.

Figure 2-5 shows the interrupt generation mechanism in DWC_mipi_i3c.

Figure 2-5 Interrupt Mechanism

2.6 Master Functionality with DWC_mipi_i3c

2.6.1 Overview of Master Role in DWC_mipi_i3c

MIPI I3C advocates a hierarchical bus architecture, where only one Master (Current Master) controls the I3C Bus at any given time. It is mandatory to have a Main Master, and optional to have Secondary Masters on the I3C Bus.

The procedure of a Master transforming to Current Master is based on the Bus Topology:

- Single Master system – Main Master is always the Current Master because it is the only Master in a Single Master system.
- Multi-Master system – During the bus initialization, Main master is the Current Master; and afterwards, any Secondary Master can become the Current Master based on the Master Request and mastership hand-off mechanism.



Note DWC_mipi_i3c, if configured as the Main Master (when register bit DEVICE_ADDR[DYNAMIC_ADDR_VALID] is set to 1), owns the DWC_mipi_i3c Bus after power-up. This means that DWC_mipi_i3c drives the SCL line and initiates transfer on the SDA line.

2.6.2 Dynamic Address Assignment (DAA)

2.6.2.1 Overview of DAA

DWC_mipi_i3c, configured as Main Master, is responsible for performing a Dynamic Address Assignment (DAA) procedure, to provide a unique Dynamic Address to each I3C device connected to the I3C Bus.

The Main Master provides a Dynamic Address to a device for the following conditions:

1. Upon initialization of the I3C Bus, and
2. When the device is connected to an already configured I3C Bus (indicated through Hot-Join or other methods)

Once a device receives a Dynamic Address, this assigned Dynamic Address is used in all the device's subsequent transactions on the I3C Bus.

2.6.2.2 Description of DAA Methods

2.6.2.2.1 SETDASA Procedure (Format 1: Primary)

The Main Master assigns Dynamic Address to any I3C devices with known Static Address using the directed command code – *Set Dynamic Address from Static Address* (SETDASA). This is faster than using the ENTDAAS procedure (described later in this section). The SETDASA process is initiated by the Main Master after cold power-up of the system.

The following flow describes the Dynamic Address Assignment using SETDASA procedure:

1. The application initializes the DWC_mipi_i3c controller.
2. The application programs the Device Address Table with the data structure as shown.

Do not use the Reserved fields during address assignment procedure. Doing so may result in unexpected behavior.



The number of locations to be programmed is decided by the following:

- a. Number of I3C Static Address devices to be assigned with Dynamic Address with single Address Assignment Command.
- b. Number of locations available in the Device Address Table.
3. The application issues the Address Assignment Command with the following:
 - a. SETDASA command code in the CMD field
 - b. DEV_INDEX field pointing the location in the Device Address Table from where the Dynamic Address is to be assigned
 - c. DEVICE_COUNT field indicates the number of devices to be assigned with the Dynamic Address by incrementing the DEV_INDEX field of the same command sequentially as many times as DEVICE_COUNT. Program the DEVICE_COUNT value in such a way that DEVICE_COUNT+DEV_INDEX should not cross the DAT table depth.
4. The DWC_mipi_i3c controller starts executing the SETDASA CCC transfer as soon as the Address Assignment command is issued as shown:



5. The Dynamic Address assignment continues until one of the following conditions occurs.
 - a. NACK response is received for the header 0x7E/W (No I3C devices present)
 - b. NACK response is received for the Static Address (Not a Valid Static Address or the device does not exist)
 - c. DEVICE_COUNT reaches zero (End of address assignment request)
6. The controller writes the transfer complete status into the Command Response queue. The Data Length Field of Response Data Structure indicates the remaining device count, if the transfer is terminated abruptly due to NACK response from the slave.

2.6.2.2.2 SETDASA Procedure (Format 2: Point-to-point)

The SETDASA CCC may also be specially used for simple point-to-point communication in I3C Minimal Bus use cases. An I3C Minimal Bus is an I3C Bus with one I3C Master Device and one I3C Slave Device.

In this special usage of the SETDASA CCC, the Master Device uses the fixed value 7'h01 as the Static Address, and uses the fixed (and reserved) value of 7'h01 as the Dynamic Address.

The following flow describes the Dynamic Address Assignment using SETDASA procedure in point-to-point communication:

1. The application initializes the DWC_mipi_i3c controller.
2. The application programs the Device Address Table with the data structure as follows:

Do not use the Reserved fields during address assignment procedure. Doing so may result in unexpected behavior.



- The number of locations to be programmed is 1 as there is only one Slave in the system to assign the address.
- The Static address and the Dynamic address field of the data structure must be programmed as 7'h01.

3. The application issues the Address Assignment Command with the following:
 - a. SETDASA command code in the CMD field.
 - b. DEV_INDEX field pointing the location in the Device Address Table in which the static address (7'h01) and Dynamic Address (7'h01) are mentioned to be assigned.
 - c. DEVICE_COUNT field is programmed to '1' as there is only one device to be assigned with the Dynamic address.

4. The DWC_mipi_i3c controller starts executing the SETDASA CCC transfer as soon as the Address Assignment command is issued as shown:



5. The Dynamic Address assignment continues until one of the following conditions occur:
- NACK response is received for the header 0x7E/W (No I3C devices present).
 - NACK response is received for the Static Address (Not a Valid Static Address or the device does not exist).
 - DEVICE_COUNT reaches zero (End of address assignment request).
6. The controller writes the transfer complete status into the Command Response queue. The Data Length Field of Response Data Structure indicates remaining device count in case if the transfer is terminated abruptly due to NACK response from the slave.

2.6.2.3 ENTDAA Procedure

The Main Master assigns Dynamic Address to any I3C device not assigned earlier with Dynamic Address, using the broadcast command code – *Enter Dynamic Address Assignment* (ENTDAA). This process is initiated by the Main Master after cold power-up, after completing the dynamic address assignment for static address devices using SETDASA procedure or if a hot-join request is received from any I3C device. The Main Master may choose to use a single or multiple ENTDAA Address Assignment command to assign a Dynamic Address for all the connected I3C devices. The number of devices that can be assigned to a single command is limited by the configured value of the Device Address Table and the Device Characteristics Table.

The following flow describes the Dynamic Address Assignment using ENTDAA procedure:

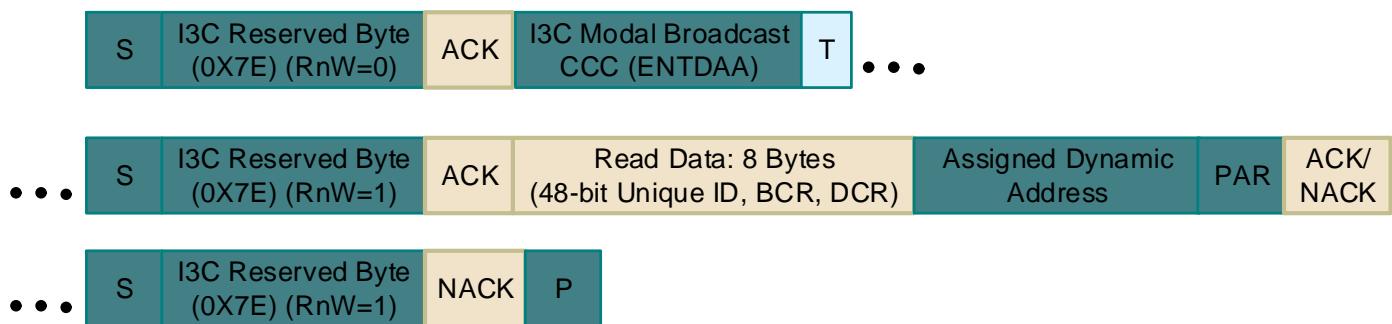
- The application initializes the DWC_mipi_i3c controller.
- The application programs the Device Address Table with the data structure as shown:
Do not use the Reserved fields during address assignment procedure. Doing so may result in unexpected behavior.

31	30	24	23	22	16	0
0	RESERVED		Dynamic Address Parity (~XOR(DYN_ADDR))	Dynamic Address (DYN_ADDR)		RESERVED

The number of locations to be programmed is decided by:

- Number of I3C devices to be assigned with Dynamic Address with single Address Assignment Command
 - Number of locations available in the Device Address Table
3. The application issues the Address Assignment Command with the following:

- a. ENTDAA command code in the CMD field.
- b. DEV_INDEX field pointing the location in the Device Address Table from where the Dynamic Address is to be assigned.
- c. DEVICE_COUNT field indicating the number of devices to be assigned with the Dynamic Address with the single command. DEVICE_COUNT value cannot be more than the IC_DEV_CHAR_TABLE_BUF_DEPTH/4, and the value should be programmed such that DEVICE_COUNT+DEV_INDEX should not exceed the DAT table depth.
4. The DWC_mipi_i3c controller starts executing the ENTDAA CCC transfer as soon as the Address Assignment command is issued as shown:



5. The first winning device gets the first Dynamic Address pointed by the DEV_INDEX of the command. The second winning device gets the second Dynamic Address pointed by the DEV_INDEX+1, and so on. The received 48-bit PID, BCR and DCR along with the assigned Dynamic Address are captured in the Device Characteristics Table.
6. The Dynamic Address assignment continues until one of the following conditions occurs:
 - a. NACK response is received for the header 0x7E/W (No I3C devices present)
 - b. NACK response is received for the restart header 0x7E/R (All devices got the Dynamic Address)
 - c. NACK response received for the Assigned Dynamic Address (Error detected by the receiving I3C device on the Dynamic Address)
 - d. DEVICE_COUNT reaches zero regardless of the number of devices yet to be assigned to the Dynamic Address (End of address assignment request)
7. The DWC_mipi_i3c controller writes the transfer complete status into the Command Response queue. The Data Length Field of Response Data Structure indicates remaining device count in case if the transfer is terminated abruptly due to NACK response from the slave.



Note Complete ENTDAA transfer from START condition to STOP condition is generated in I3C Open-Drain mode.

2.6.2.4 SETAASA Procedure

The current Master can also request for all connected Slaves (which has static address) to use their known static address as their dynamic address. This is the fastest method to assign I3C dynamic addresses to all

Slaves with static addresses. This application must use regular transfer command instead of address assignment command to issue SETAASA transfer command.

The following flow describes the dynamic address assignment using SETAASA procedure:

1. The application initializes the DWC_mipi_i3c controller.
2. The application issues the Regular Transfer Command to generate Broadcast SETAASA CCC Transfer without Data Payload.
3. The DWC_mipi_i3c controller starts executing the SETAASA CCC transfer as soon as the Regular Transfer command is issued, as shown.



4. The Dynamic Address assignment continues until it generates SETAASA CCC on the line or it ends if it receives NACK response for the header 0x7E/W (No I3C devices present).
5. The controller writes the transfer complete status in to the Command Response queue. The ERR_STS field indicates whether the transfer is completed successfully, or it has terminated abruptly due to NACK response from the slaves.



Note After successfully transmitting SETAASA CCC command, the application should update the DAT [Dynamic Address] to DAT [Static Address] to reflect that the dynamic address is equivalent to static address of Slave.

2.6.3 In-Band Interrupt (IBI) Detection and Handling

This chapter describes the In-Band Interrupt detection and handling mechanism of the host controller. The I3C Slave devices can initiate communication to the current Master through In-Band Interrupt (IBI). The following types of IBIs are possible on an I3C bus.

- Hot-Join Request (HJ) from a Hot-Join capable Slave
- Slave Interrupt Request (SIR) from a Slave
- Master Ownership Request (MR) from a Master capable Slave (Secondary Master)

Enable the controller to allow the host controller to detect (SDA low) and receive the In-Band interrupt (IBI ID). This enables the host controller to start providing the SCL clocks (period = I3C_OD_LCNT+I3C_OD_HCNT) to receive the IBI ID from the requesting Slave device. The host controller detects the In-Band interrupt in the following scenarios:

- Upon detecting low on the SDA input port after a Power-On-Reset (POR).
- Upon detecting an arbitration loss during an address phase of any master-initiated transfer following a START condition (not RESTART).
- Upon detecting the SDA input port going (not initiated by the controller) low following a STOP condition (Slave Initiated IBI).

Table 2-1 captures the key controls that are available in Master mode of operation which decides the response to the Slave-initiated IBIs and notification to the Master application on any rejected IBI (NACK).

Table 2-1 IBI Response and Notify Controls

	Device Role (Configuration)				Slave Only	Slave -Lite
	Master Only	HCI Master	Secondary Master		Slave Only	Slave -Lite
Controller Enable	DEVICE_CTRL [ENABLE]	HC_CONTROL [ENABLE]	DEVICE_CTRL [ENABLE]		DEVICE_CTRL [ENABLE]	NA
MR Response Control (to Slave)	DAT Entry [MR_REJECT]	DAT Entry [MR_REJECT]	IBI_MR_REQ_REJECT Register	NA	NA	
SIR Response Control (to Slave)	DAT Entry [SIR_REJECT]	DAT Entry [SIR_REJECT]	IBI_SIR_REQ_REJECT Register	NA	NA	
HJ Response Control (to Slave)	DEVICE_CTRL [HOT_JOIN_CTRL]	HC_CONTROL [HOT_JOIN_CTRL]	DEVICE_CTRL [HOT_JOIN_CTRL]	NA	NA	
MR Reject Notify Control (to APP)	IBI_QUEUE_CTRL [NOTIFY_MR_REJECTED]	IBI_NOTIFY_CTRL [NOTIFY_MR_REJECTED]	IBI_QUEUE_CTRL [NOTIFY_MR_REJECTED]	NA	NA	
SIR Reject Notify Control (to APP)	IBI_QUEUE_CTRL [NOTIFY_SIR_REJECTED]	IBI_NOTIFY_CTRL [NOTIFY_SIR_REJECTED]	IBI_QUEUE_CTRL [NOTIFY_SIR_REJECTED]	NA	NA	
HJ Reject Notify Control (to APP)	IBI_QUEUE_CTRL [NOTIFY_HJ_REJECTED]	IBI_NOTIFY_CTRL [NOTIFY_HJ_REJECTED]	IBI_QUEUE_CTRL [NOTIFY_HJ_REJECTED]	NA	NA	
IBI Payload Control (SIR)	DAT Entry [IBI_PAYLOAD]	DAT Entry [IBI_PAYLOAD]	NA	NA	NA	
SIR Data Chunk Size	QUEUE_THLD_CTRL [IBI_DATA_THLD]	QUEUE_THLD_CTRL [IBI_DATA_THLD]	NA	NA	NA	

2.6.3.1 I3C Hot-Join (HJ) Request

If the received IBI ID matches with the Hot-Join ID (7'b00000010, RnW=0), then the response for the received HJ request is based on the programmed HJ Response Control.

If the HJ Response Control is set to 0, then the controller responds to the HJ request with ACK and sets 'IBI_STS' field to ACK to indicate to the application that the controller has ACK'd the received HJ request.

If the HJ Response Control is set to 1, then the controller responds to the HJ request with a NACK followed by issuing a broadcast DISEC CCC command (DISHJ bit set) with the RESTART condition. This disables the HJ request generation from all the unaddressed devices at that instant. The controller sets the 'IBI_STS' field to NACK to indicate to the application that the controller has rejected the received HJ request. The

application can optionally set the HJ Reject Notify Control to get an IBI Status for a rejected HJ request. Otherwise, the controller moderates the IBI Status generation for rejected HJ requests.

Based on the IBI Status for the HJ request, the application must issue the ENTDAA to assign the dynamic address for the devices that generates the HJ request.

2.6.3.2 I3C Master Request

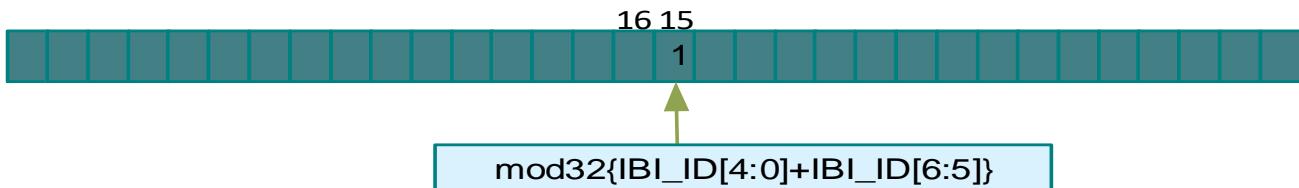
If the received IBI ID matches with the Dynamic Address of any one of the valid entries in the DAT and the received RnW bit is 0, then the response for the received MR is based on the programmed MR Response Control.

The Secondary Master role allows the application to either set the MR Response Control to 0 or 1 individually to accept or reject the MR request respectively. The application is expected to be always set to 1, the MR Response Control for the Master Only and HCI Master configuration. This requirement allows the Master to reject any MR request if received unexpectedly from a malfunctioning Slave device.

2.6.3.2.1 MR Response Control

- In the Secondary Master configuration, the controller handles the MR response to the targeted slave through a 32-bit vector control register (IBI_MR_REQ_REJECT). This register is used to control the response individually for each I3C Master capable slave devices. Each bit field of the control register is mapped to a unique 7-bit slave device address through the modular arithmetic operation as shown in [Figure 2-6](#).

Figure 2-6 MR IBI Response Control



The sum of the lower 5 bits and the upper 2 bits of the Master Request ID (Dynamic Address of the requesting slave) is wrap around upon reaching the value 32 (module 32) to uniquely map the bit position of the 32-bit control register.

- In the Master Only and HCI Master configurations, the controller handles the MR response to the targeted slave through the MR_REJECT control of the corresponding DAT entry.

If the MR Response Control is set to 1, then the controller responds to the MR request with NACK followed by issuing a directed DISEC CCC command (DISMR bit set) with the RESTART condition to the MR initiated slave. The controller sets the 'IBI_STS' field as NACK to indicate to the application that the controller has rejected the received MR request. The application can optionally set the MR Reject Notify Control to get an IBI Status for a rejected MR request. Otherwise, the controller moderates the IBI Status generation for rejected MR requests.

If the MR Response Control is set to 0, then the controller responds to the MR request with ACK and sets 'IBI_STS' field to ACK, which indicates the application that the controller has ACK'd the received MR request.

Based on the IBI Status for the MR request, the application must follow the Master ownership handover procedure.



In Master only and HCI Master configurations, if the received IBI ID (MR) does not match with any of the dynamic address entry of the DAT table, then the controller sends a NACK response and generates IBI status to the application. The controller does not send any disable event command and expects the application to take necessary action if the unknown device repeatedly interrupts. In this case, the generation of IBI status is independent of the MR Reject Notify Control settings.

2.6.3.3 I3C Slave Interrupt Request (SIR)

If the received IBI ID matches with the Dynamic Address of any one of the valid entries in the DAT and the received RnW bit is 1, then the response for the received SIR is based on the programmed SIR Response Control.

[Table 2-2](#) captures the supported SIR features in different configurations.

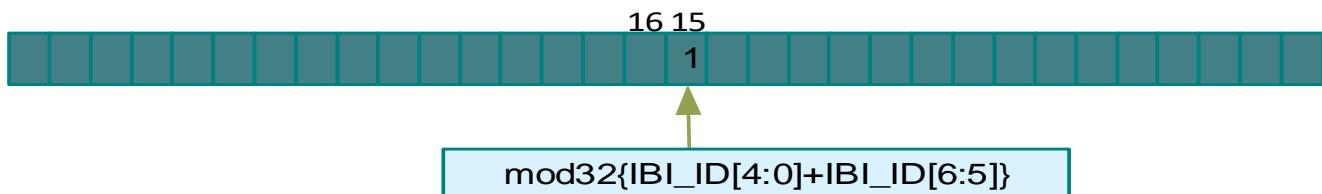
Table 2-2 Slave Interrupt Request Feature Support

	Device Role (Configuration)				
	Master Only	HCI Master	Secondary Master	Slave Only	Slave-Lite
SIR without Payload	Supported	Supported	Supported	Supported	Supported
SIR with Payload	Supported	Supported	Not Supported	Not Supported	Supported
SIR Auto-command	Not Supported	Supported (SDR Only)	Not Supported	NA	NA

2.6.3.3.1 SIR Response Control:

1. In the Secondary Master configuration, the controller handles the SIR response to the targeted Slave through a 32-bit vector control register (IBI_SIR_REQ_REJECT). This register is used to control the response individually for each I3C slave device. Each bit field of the control register is mapped to a unique 7-bit Slave device address through the modular arithmetic operation as shown in [Figure 2-7](#).

Figure 2-7 SIR IBI Response Control



The sum of the lower 5 bits and the upper 2 bits of the Slave Interrupt Request ID (Dynamic Address of the requesting slave) is wrap around upon reaching the value 32 (module 32) to uniquely map the bit position of the 32-bit control register.

2. In the Master only and HCI Master configurations, the controller handles the SIR response to the targeted Slave through the SIR_REJECT control of the corresponding DAT entry.

If SIR Response Control is set to 1, then the controller NACK's the SIR and then issues a directed DISEC CCC command (DISINT bit set) with the RESTART condition targeting the matching Dynamic Address. This disables the SIR generation in the requested Slave device. The IBI status for the corresponding SIR indicates the application that the incoming IBI is NACK'd through 'IBI_STS' descriptor.

If SIR Response Control is set to 0, then the controller ACK's the SIR. If the IBI Payload Control is set to 1, then the controller continues to generate the SCL clocks after the ACK to accept the IBI payload bytes starting from the MDB until the slave device terminates the transfer. The application must set the IBI Payload Control to 1 only when the Slave device supports the mandatory byte (BCR[2]=1). If the IBI Payload Control is set to 0, then the controller stops generating the SCL clock after acknowledging the SIR. The controller then notifies the application to take necessary action for the accepted SIR. The application must not set the IBI Payload Control bit to 1 when the Slave device does not support the mandatory data byte (BCR[2]=0).



Note In Master only and HCI Master configurations, if the received IBI ID (SIR) does not match with any of the dynamic address entry of the DAT table, then the controller sends a NACK response and generates IBI status to the application. The controller does not send any disable event command and expects the application to take necessary action if the unknown device repeatedly interrupts. In this case, the generation of IBI status is independent of the SIR Reject Notify Control settings.

The HCI Master configuration supports the auto-command (read) feature. The controller issues a read command to the interrupting device (SIR) upon detecting an expected MDB value. The expected MDB value is set in the DAT by programming DAT[AUTOCMD_MASK] and DAT[AUTOCMD_VALUE] fields. The controller computes the match as per the following expression before issuing a read command to the requesting device:

DAT[AUTOCMD_MASK] & MDB = DAT[AUTOCMD_VALUE]

This feature enables the application to read the Slave as soon as the data is available for reading (indicated through SIR) and thereby avoiding any delay in issuing a read command due to system latency.

If there is a match on the received MDB, the controller accepts the payload until the Slave device terminates and then issue a Read command to the same Slave with the RESTART condition. If there is no match on the received MDB, the controller does not issue the auto command and waits for the Slave device to terminate the IBI transfer.



Note Auto Command is issued only in SDR Mode and is not supported in HDR Modes

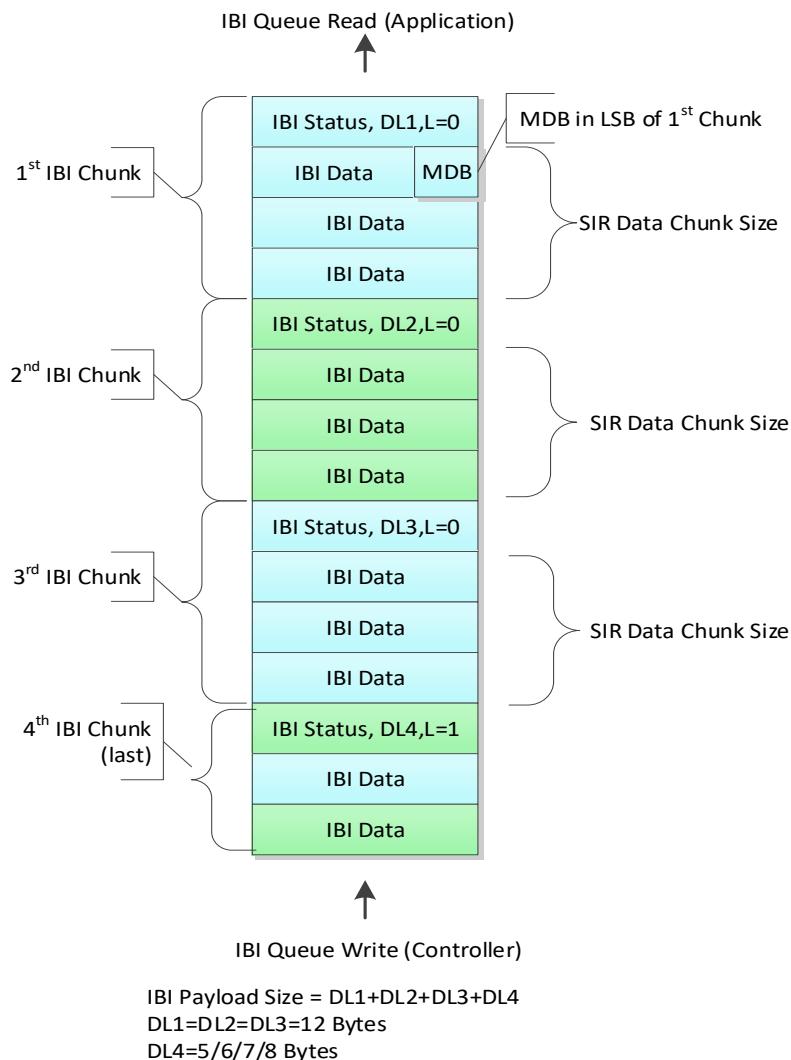
2.6.3.4 IBI Queue Data Structure

The IBI queue comprises of the interleaved data (If IBI Payload Control is supported) received from the Slave device, and the status generated from the controller for the respective data slice of SIR or HJ/MR. For the configurations that supports IBI without payload, only IBI status entry is written in to the IBI queue.

The data portion of the structure comprises of either the data received from the Slave device along with the IBI (MDB + Other payload bytes if any) or the data received for Auto-Command (if supported) issued for

the requesting device. If the data payload size of the SIR goes above the programmed SIR Data chunk size, then the controller slices the incoming data bytes into multiple chunks and generates an IBI status for each chunk. The last data chunk is indicated as LAST_STS=1 in the corresponding IBI status. [Figure 2-8](#) shows an example of the IBI queue data structure holding the IBI data (SIR) and the status for an IBI with non-multiple chunk size data.

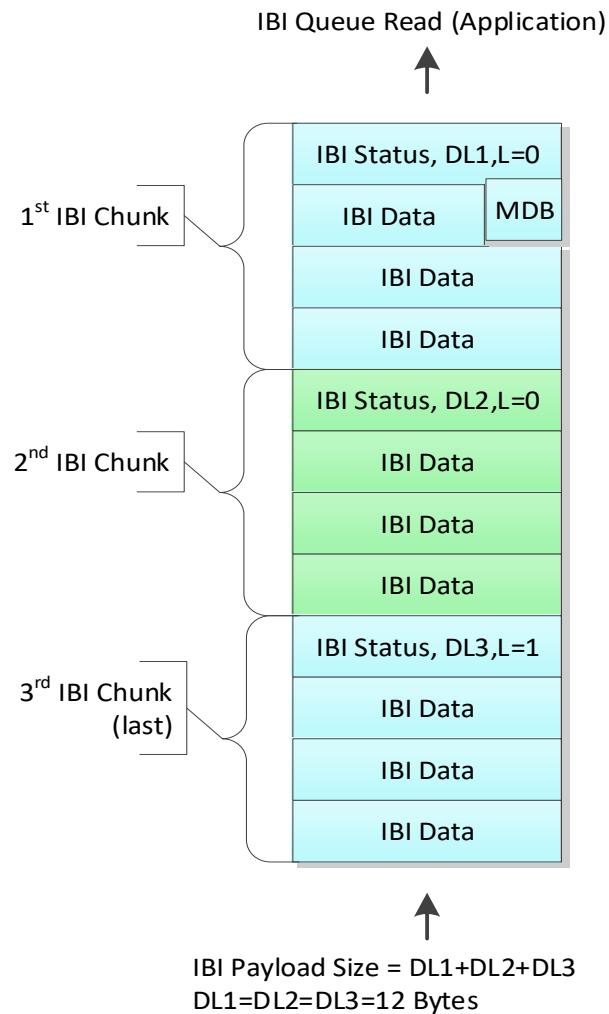
Figure 2-8 IBI with Payload Structure (with last data chunk less than programmed data chunk size)



The application reads the IBI queue upon detecting the interrupt status `PIO_INTR_STATUS[IBI_STATUS_THLD_STAT]` (HCI Master Configuration) or `INTR_STATUS[IBI_THLD_STS]` (Master Only and Secondary Master Configuration). The interrupt can be moderated by setting the `QUEUE_THLD_CTRL[IBI_STATUS_THLD]` field. The first location read always provides the IBI status for the 1st chunk, which includes the data length of that chunk in bytes. The application can read the data portion of the chunk without waiting for any further interrupt. The number of data reads must be limited to the data length field of the corresponding status.

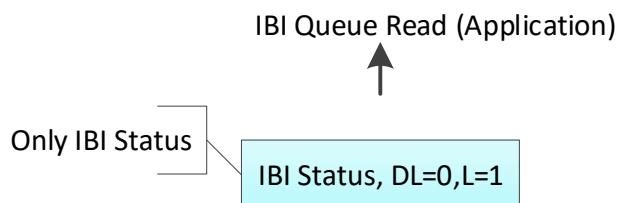
[Figure 2-9](#) shows an example of the IBI queue data structure holding the IBI data and the status for an IBI with exact multiple of programmed SIR Data Chunk Size.

Figure 2-9 IBI with Payload Structure (with the last chunk equals to the programmed chunk size)



[Figure 2-10](#) shows an example of the IBI queue data structure holding only the IBI status. This structure is applicable for all the IBIs that does not support the payload, which includes the HJ, the SIR without payload, and the MR.

Figure 2-10 IBI without Payload Structure



2.6.3.5 IBI Notification Control

Normally, the controller generates an IBI response only for the received IBIs for which ACK response is sent. Optionally, you can enable the response generation for the NACK'ed IBIs by setting the appropriate Reject Notify Control (Refer [Table 2-1](#)).

When the Hot-Join Reject Notify Control is set to 1, the controller notifies the application by generating an IBI response when a valid Hot-Join request is received for which a NACK response was sent followed by a broadcast Auto disable CCC command.

When the MR Reject Notify Control is set to 1, the controller notifies the application by generating an IBI response when a valid Master request is received, for which a NACK response is sent followed by a directed Auto disable CCC command to the requesting Slave device.

When the SIR Reject Notify Control is set to 1, the controller notifies the application by generating an IBI response when a valid SIR request is received, for which a NACK response is sent followed by a directed Auto disable CCC command to the requesting Slave device.



1. When an unsupported/Invalid IBI (MR, HJ/RnW=1) or an IBI from an unknown Slave device (no valid entry in DAT) is received, the controller responds with NACK regardless of Reject Notify Control and notifies the application. The controller does not issue any Auto disable command in this case.
2. If the IBI Status Queue is full when any non-SIR request is detected, the controller continues to provide the SCL clock to receive the IBI ID and responds with NACK at the end (But does not issue DISEC CCC). This allows the requesting Slave to re-attempt the IBI on the next START condition or upon detecting a bus available period
3. If the IBI Data Queue is full when an SIR request is detected, the controller continues to provide SCL clock to receive the IBI ID and enters to clock stall state during the ACK space of the IBI ID until a free space is available in the IBI Data Queue. If the IBI Data Queue goes full in the middle of an SIR data reception (ACK'ed IBI), the controller stalls the SCL clock by driving low until a free space is available to start accepting further bytes.
4. To avoid IBI Status and Data Queues becoming full, it is important to drain Status and Data from IBI queue as soon as they become available. Hence, set the IBI Status threshold to be 0. This ensures that as soon as the first status from an IBI is available in the IBI queue, the application is notified through IBI Status Threshold Met interrupt.

2.6.4 Disabling DWC_mipi_i3c Master

The application can disable the DWC_mipi_i3c when operating as a Master at any time by clear the DEVICE_CTRL[ENABLE] bit. The application should then poll the DEVICE_CTRL[ENABLE] bit for until it turns to 1'b0 for confirming that the controller is in disabled state. If the Master controller is busy in executing any I3C bus transfers (like receiving/transmitting an I3C transfer or receiving an IBI), then the controller enters the disabled state only after completing either the transfer command with TOC=1 or the IBI reception. Once the controller enters the disabled state, it does not execute any commands from the Command Queue and does not provide any clock for a new IBI. The application is expected to flush/drain all the queues and the FIFOs before re-enabling the controller.

2.6.5 Aborting Transfers of DWC_mipi_i3c Master

The application of the Master can request the controller to abort any ongoing I3C bus transfer by setting DEVICE_CTRL[ABORT] bit.

In response to an abort request, the controller issues the STOP condition after the on-going data byte is transferred or received. The controller then generates an interrupt, sets the INTR_STATUS[TRANSFER_ABORT_STAT] bit and enters the halt state.

The controller then waits for the application to issue the resume command by setting the DEVICE_CTRL[RESUME] bit to exit the halt state. The application is expected to flush/drain all the queues and the FIFOs before programming the DEVICE_CTRL/HC_CONTROL[RESUME] bit to resume the controller.

2.6.6 Master Data Structures in Non-HCI Mode

This chapter describes the Master data structures of DWC_mipi_i3c in non-HCI mode. This chapter consists of the following sections:

- “Command Data Structure”
- “Response Data Structure”
- “IBI Status and Data Structure”
- “Device Address Table Data Structure”
- “Device Characteristics Table Data Structure”

2.6.6.1 Command Data Structure

The command port of DWC_mipi_i3c accepts the following four types of data structures to initiate a transfer on the I3C Bus:

- Transfer Command Data Structure
- Transfer Argument Data Structure
- Short Data Argument Data Structure
- Address Assignment Command Data Structure

If a transfer consists of payload of at least one byte, then either the Transfer Argument or the Short Data Argument must be written into the command port prior to the Transfer Command.

Figure 2-11 DWC_mipi_i3c Command Structure

BIT_OFFSET CMD_TYPE	31	30	29	28	27	26	25	24	23 : 21	20 : 16	15	14 : 8	7	6	5 : 3	2 : 0
Transfer Command	PEC	TOC	RESV	RnW	SDAP	ROC	DBP	RESV	SPEED	DEV_INDX	CP	CMD	CMD	TID	CMD_ATTR (0)	
Transfer Argument	DATA_LENGTH										DB	RESV			CMD_ATTR (1)	
Short Data Argument	DATA_BYTE_3						DATA_BYTE_2			DATA_BYTE_1 /DB	RESV	BYTE_STRB	CMD_ATTR (2)			
Address Assignment command	RESV	TOC	RESV		ROC	DEV_COUNT		DEV_INDX	RESV	CMD	CMD	TID	CMD_ATTR (3)			

2.6.6.1.1 Transfer Command Data Structure

In the Master mode of operation, the Transfer Command is used to initiate CCC and private transfers. For transfers with data payload, additional data structure (either Transfer Argument or Short Data Argument) is used to provide payload details.

Table 2-3 Transfer Command Data Structure

SL.NO	FIELD NAME	BIT-FIELD	WIDTH	DESCRIPTION
1	CMD_ATTR	2:0	3	<p>Command Attribute Defines the Command Type and its Bit-Field Format.</p> <ul style="list-style-type: none"> ■ 0: Transfer Command ■ 1: Transfer Argument ■ 2: Short Data Argument ■ 3: Address Assignment Command ■ 4-7: Reserved
2	TID	6:3	4	<p>Transaction ID This Field is used as the identification tag for the commands. The I3C controller returns this ID along with the response upon completion or upon error.</p> <ul style="list-style-type: none"> ■ 4'b0000 - 4'b0111 - User-Defined TID ■ 4'b1000 - 4'b1111 - Reserved for I3C controller.
3	CMD	14:7	8	<p>Transfer Command This field is used to define the Transfer Command type. The field can be programmed to:</p> <ol style="list-style-type: none"> 1. 8-bit Common Command Code for CCC transfers. 2. 7-bit Command Code for HDR-TS or HDR-DDR transfers (bit[14] is reserved).
4	CP	15	1	<p>Command Present This bit is used to control whether the transfer should be initiated with the Transfer Command represented in the 'CMD' field or not.</p> <ul style="list-style-type: none"> ■ 0 - CMD field is not valid ■ 1 - CMD field is valid <p>This bit is applicable for CCC and HDR transfers.</p>
5	DEV_INDX	20:16	5	<p>Device Index This field is used to refer the Device Address Table for getting the target address. DEV_INDX field points to the offset address of Device Address Table.</p>

Table 2-3 Transfer Command Data Structure (Continued)

SL.NO	FIELD NAME	BIT-FIELD	WIDTH	DESCRIPTION
6	SPEED	23:21	3	<p>Speed This field is used to indicate the speed in which the transfer should be driven.</p> <p>Values (I3C Mode):</p> <ul style="list-style-type: none"> ■ 0: SDR0 ■ 1: SDR1 ■ 2: SDR2 ■ 3: SDR3 ■ 4: SDR4 ■ 5: HDR-TS ■ 6: HDR-DDR ■ 7: I2C FM <p>Note that HDR-DDR and HDR-TS are supported only if the Transfer Command Structure is set-up to use Transfer Argument and NOT Short Data Argument.</p> <p>Values (I2C mode):</p> <ul style="list-style-type: none"> ■ 0: I2C FM ■ 1: I2C FM+ ■ 2 -7: Reserved
7	RESV	24	1	Reserved
8	DBP	25	1	<p>Defining Byte Present DBP indicates whether the current CCC command is with Defining Byte or not. This bit field is valid only when CP (command Present) bit is enabled, otherwise this bit is ignored.</p> <ul style="list-style-type: none"> ■ 0: Defining Byte is not present. ■ 1: Defining Byte is present. <p>This bit is applicable only for Broadcast and Directed SDR CCC Transfers.</p>

Table 2-3 Transfer Command Data Structure (Continued)

SL.NO	FIELD NAME	BIT-FIELD	WIDTH	DESCRIPTION
9	ROC	26	1	<p>Response On Completion</p> <p>This field indicates whether the Response Status is required or not, after the execution of this command for the successful transfer.</p> <ul style="list-style-type: none"> ■ 1 - Response Status is required. ■ 0 - Response Status is not required. <p>Note:</p> <p>1) The exception to the control is that the response status gets generated when the transfer has encountered an error condition.</p> <p>2) It is recommended that the ROC bit is always set to 1 for the Read commands (RnW=1), so that the number of data received is indicated (through DATA_LENGTH field in Response port) if the Slave terminates early than the Master.</p>
10	SDAP	27	1	<p>Short Data Argument Present</p> <p>This field indicates whether the command written prior to the Base command should be treated as Short Data Argument or the Transfer Argument.</p> <ul style="list-style-type: none"> ■ 0 - Prior written command is Transfer Argument. ■ 1 - Prior written command is Short Data Argument.
11	RnW	28	1	<p>Read and Write</p> <p>This bit controls whether a Read or Write transfer is performed.</p> <ul style="list-style-type: none"> ■ 0 - Write Transfer ■ 1 - Read Transfer <p>Note: In HDR transfers, this bit is used to set the Read/Write flag of the HDR-TS/HDR-DDR Command Code.</p>
12	RESV	29	1	Reserved
13	TOC	30	1	<p>Termination On Completion</p> <p>This bit controls whether a STOP need to be issued after the completion of the transfer or not.</p> <ul style="list-style-type: none"> ■ 1 - STOP issued after this transfer. ■ 0 - The next transfer starts with RESTART condition.

Table 2-3 Transfer Command Data Structure (Continued)

SL.NO	FIELD NAME	BIT-FIELD	WIDTH	DESCRIPTION
14	PEC	31	1	<p>Parity Error Check Enable This bit enables generation and validation of PEC byte for SDR CCC and private transfers.</p> <ul style="list-style-type: none"> ■ 0: PEC check is disabled. ■ 1: PEC check is enabled. <p>Note: This bit is valid only for SDR Transfers and not for HDR Transfers.</p>

2.6.6.1.2 Transfer Argument Data Structure

Transfer Argument Data Structure is used to provide payload related information. This data structure must be used when the external DMA is used to get the payload bytes.

Table 2-4 Transfer Argument Data Structure

SL.NO	FIELD NAME	BIT-FIELD	WIDTH	DESCRIPTION
1	CMD_ATTR	2:0	3	<p>Command Attribute Defines the Command Type and its Bit-Field Format.</p> <ul style="list-style-type: none"> ■ 0: Transfer Command ■ 1: Transfer Argument ■ 2: Short Data Argument ■ 3: Address Assignment Command ■ 4-7: Reserved
2	RESV	7:3	5	Reserved
3	DB	15:8	8	<p>Defining Byte Value DB indicates the 8-bit defining byte to be transferred in the CCC transfer. This byte is valid only when both CP and DBP bits are enabled, otherwise the controller ignores this byte.</p>
4	DL	31:16	16	<p>Data Length This field is used to indicate the Data length of the transfer. For CCC transfers, it is expected that the correct length be programmed. The Master controller deals with all CCCs transparently.</p>

2.6.6.1.3 Short Data Argument Data Structure

Short Data Argument Data Structure is used to provide payload related information for the TX transfers when the payload size is less than or equal to 3 bytes.

Table 2-5 Short Data Argument Data Structure

SL.NO	FIELD NAME	BIT-FIELD	WIDTH	DESCRIPTION
1	CMD_ATTR	2:0	3	<p>Command Attribute Defines the Command Type and its Bit-Field Format.</p> <ul style="list-style-type: none"> ■ 0: Transfer Command ■ 1: Transfer Argument ■ 2: Short Data Argument ■ 3: Address Assignment Command ■ 4-7: Reserved
2	BYTE_STRB	5:3	3	<p>Byte Strobe This Field is used to select the valid data bytes of the Short Data Argument.</p> <ul style="list-style-type: none"> ■ BYTE_STRB[0] - Data Byte -0 Valid Qualifier ■ BYTE_STRB[1] - Data Byte -1 Valid Qualifier ■ BYTE_STRB[2] - Data Byte -2 Valid Qualifier <p>Valid combinations = 3'b001, 3'b011 and 3'b111</p>
3	RESV	7:6	2	Reserved
4	DATA_BYTE_0/ DB	15:8	8	<p>Data Byte -0/Defining Byte This field is used for storing the Data Byte -0. DB indicates the 8-bit Defining Byte to be transferred in the CCC transfer. This byte is valid only when both CP and DBP bits are enabled, otherwise the controller ignores this byte.</p> <p>Note: When Defining Byte (DBP) is enabled in Short Data Argument,</p> <ol style="list-style-type: none"> 1. The BYTE_STRB[0] bit is treated to be always enabled whether or not you set it. 2. The maximum possible payload bytes to be sent is '2' due the first byte being occupied by the Defining Byte.
5	DATA_BYTE_1	23:16	8	<p>Data Byte -1 This field is used for storing the Data Byte -1.</p>
6	DATA_BYTE_2	31:23	8	<p>Data Byte -2 This field is used for storing the Data Byte -2.</p>



Note Short Data Argument is not supported for HDR transfers because in HDR modes, a maximum of two words can be transferred in this format.

2.6.6.1.4 Address Assignment Command Data Structure

In the Master mode of operation, the Address Assignment Command is used to initiate ENTDAA and SETDASA transfers.

Table 2-6 Address Assignment Command Data Structure

SL.NO	FIELD NAME	BIT-FIELD	WIDTH	DESCRIPTION
1	CMD_ATTR	2:0	3	<p>Command Attribute Defines the Command Type and its Bit-Field Format.</p> <ul style="list-style-type: none"> ■ 0: Transfer Command ■ 1: Transfer Argument ■ 2: Short Data Argument ■ 3: Address Assignment Command ■ 4-7: Reserved
2	TID	6:3	4	<p>Transaction ID This Field is used as the identification tag for the commands. The I3C controller returns this ID along with the response upon completion or upon error.</p> <ul style="list-style-type: none"> ■ 4'b0000 - 4'b0111 - User-Defined TID ■ 4'b1000 - 4'b1111 - Reserved for I3C controller.
3	CMD	14:7	8	<p>Address Assignment CCC This field is used to define the Address Assignment Command type used in the transfer. This field is used for representing the ENTDAA or SETDASA Common Command codes.</p>
4	RSVD	15	1	Reserved
5	DEV_INDX	20:16	5	<p>Device Index This field is used to indicate the start pointer of the Device Table from where the Dynamic Address is to be picked and assigned to the I3C devices.</p>
6	DEV_COUNT	25:21	5	<p>Device Count This field is used to represent the number of devices to be assigned with the Dynamic Address.</p>
7	ROC	26	1	<p>Response On Completion This field indicates whether the Response Status is required or not, after the execution of this command for the successful transfer.</p> <ul style="list-style-type: none"> ■ 1 - Response Status is Required. ■ 0 - Response Status is not required. <p>Note: The exception to this control is that the response status gets generated when the transfer has encountered an error condition.</p>

Table 2-6 Address Assignment Command Data Structure (Continued)

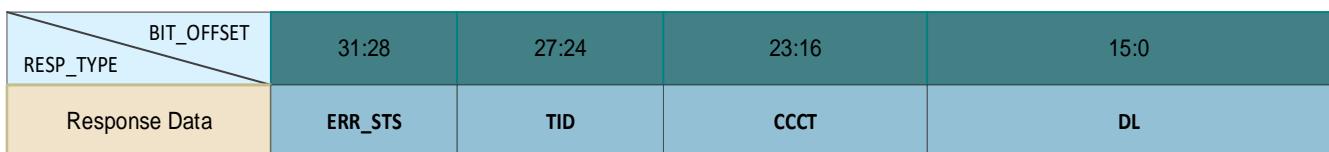
SL.NO	FIELD NAME	BIT-FIELD	WIDTH	DESCRIPTION
8	RSVD	29:27	3	Reserved
9	TOC	30	1	<p>Termination On Completion</p> <p>This field controls whether a STOP need to be issued after the completion of the transfer or not.</p> <ul style="list-style-type: none"> ■ 1 - STOP issued after this transfer. ■ 0 - The next transfer starts with RESTART condition.
10	RSVD	31	1	Reserved



Note This command is not used for the address assignment commands ‘SETAASA’ Broadcast CCC transfer and ‘SETNEWDA’ directed CCC transfer. The application has to use regular transfer command for issuing SETAASA or SETNEWDA command.

2.6.6.2 Response Data Structure

The DWC_mipi_i3c controller returns the status of the transfer command into the response queue with this data structure.

Figure 2-12 Response Data Structure**Table 2-7 Response Data Structure**

SL.NO	FIELD NAME	BIT-FIELD	WIDTH	DESCRIPTION
1	DL	15:0	16	<p>Data Length</p> <p>For Write transfers, this field represents the remaining data length of the transfer if the transfer is terminated early (remaining data length = requested data length - transferred data length)</p> <p>For Read transfers, this field represents the actual amount of data received in bytes.</p> <p>For Address Assignment command, this field represents the remaining device count.</p>

Table 2-7 Response Data Structure (Continued)

SL.NO	FIELD NAME	BIT-FIELD	WIDTH	DESCRIPTION
2	CCCT	23:16	8	<p>CCC/HDR Header Type</p> <p>This field represents the CCC type of the received vendor extension CCC packet or the HDR header of the received HDR transaction.</p> <p>This field indicates the CCC type when the TID is set to 4'b1111 (reserved for all other transactions). During the master transactions and slave non-HDR transactions, this field returns 8'b0000_0000 and can be considered as don't care.</p>
3	TID	27:24	4	<p>Transaction ID</p> <p>This Field is used as the identification tag for the commands. The I3C controller returns the ID received through commands.</p> <ul style="list-style-type: none"> ■ 4'b0000 - 4'b0111 - User-Defined TID (specified in the Transfer/Address Assignment Command) ■ 4'b1000 - Master Write Data Status (as Slave only) ■ 4'b1111 - DEFSLVS Status (as Slave only) ■ 4'b1001 - 4'b1110 - Reserved for I3C controller.
4	ERR_STS	31:28	4	<p>Error Type</p> <p>Defines the Error Type of the processed command or received vendor extension CCC packet (Slave mode).</p> <ul style="list-style-type: none"> ■ 0: No Error ■ 1: CRC Error ■ 2: Parity Error ■ 3: Frame Error ■ 4: I3C Broadcast Address NACK Error ■ 5: Address NACK'ed. This bit is set in case the Slave NACK's for Dynamic Address Assignment during ENTDAA process. ■ 6: Receive Buffer Overflow/Transmit Buffer Underflow (Only for HDR Transfers) ■ 7: Reserved ■ 8: Transfer Aborted ■ 9: I2C Slave Write Data NACK Error ■ 10-11: Reserved ■ 12: PEC Error. This bit is set if PEC byte validation error occurs in read transfers when TRANSFER_COMMAND[PEC] bit is set to 1. ■ 13-15: Reserved

2.6.6.3 IBI Status and Data Structure

The DWC_mipi_i3c controller returns the status of the received In-Band Interrupts (IBI) into the IBI queue with this data structure when IBI with Payload Configuration is not selected (IC_HAS_IBI_DATA==0).

Figure 2-13 IBI without Payload Status Structure

BIT_OFFSET RSP_TYPE	31:28	27:24	23:16	15:8	7:0
Response Data	IBI_STS	RESERVED	RESERVED	IBI_ID	DL

Table 2-8 IBI without Payload Status Structure

SL.NO	FIELD NAME	BIT-FIELD	WIDTH	DESCRIPTION
1	DATALENGTH	7:0	8	Data Length The length of the data bytes received along with the IBI.
2	IBI_ID	15:8	8	IBI Identifier The Byte received after START which includes the address and the R/W bit.
3	RESERVED	27:16	12	Reserved
4	IBI_STS	31:28	4	IBI Status <ul style="list-style-type: none"> ■ 0: Responded with ACK ■ 1: Responded with NACK ■ 2-15: Reserved

The DWC_mipi_i3c controller returns the status of the received In-Band Interrupts (IBI) into the IBI queue with this data structure when IBI with payload configuration is not selected (IC_HAS_IBI_DATA==1).

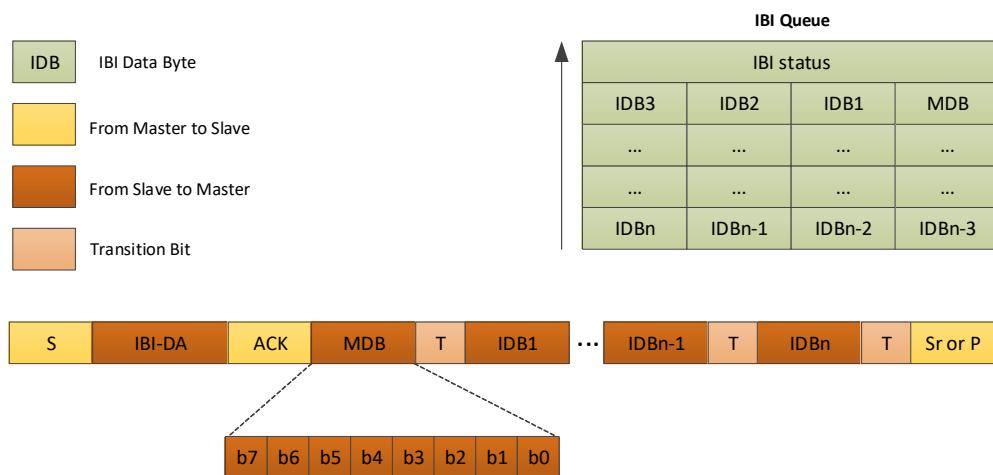
Table 2-9 IBI with Payload Status Structure

Bits	FIELD NAME	Access Type	Reset Value	DESCRIPTION
31	IBI_STS	R	0x0	IBI Status Indicates the status of the response returned for the received IBI. 1'b0: The received IBI is responded with an ACK. Any non-zero value of the DATA_LEN field indicates the presence of data payload for the ACK'ed IBI. 1'b1: The received IBI is responded with a NACK. An auto disable CCC command is issued if the received IBI address is valid and matching with the DAT entry. If an IBI is received from an unknown address (not a valid entry in DAT), the IBI_STS is set to 1.

Table 2-9 IBI with Payload Status Structure (Continued)

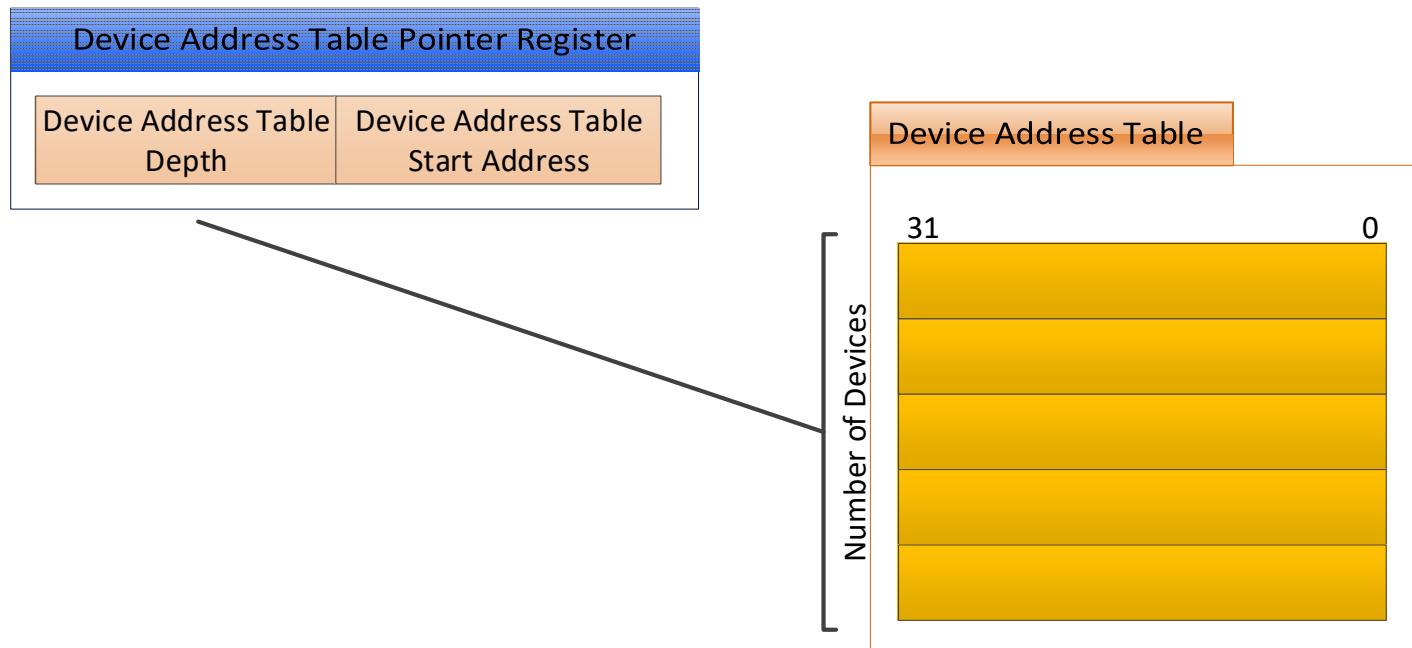
Bits	FIELD NAME	Access Type	Reset Value	DESCRIPTION
30	ERROR	R	0x0	<p>Error</p> <p>Indicates that during IBI Auto command, an error is encountered and the data from the Slave device is partially or fully discarded. The following errors are detected.</p> <ul style="list-style-type: none"> ■ CRC/Parity Error (applicable for HDR transaction) ■ Slave Address NACK ■ 0x7E Address NACK ■ IBI buffer overflow (applicable for IBI HDR modes) <p>1'b0: No Error. Transaction complete with no errors. 1'b1: Error. Error encountered during IBI Auto command phase.</p>
29:25	RESV	R	0x0	-
24	LAST_STATUS	R	0x0	<p>Last Status</p> <p>When set, indicates that this status is the last for the received IBI. If the payload of the received SIR exceeds the programmed IBI data threshold, then the controller splits the IBI payload into multiple chunks of IBI_DATA_THLD size (max) which includes the timestamp bytes if enabled.</p>
23:16	RESV	R	8'h00	Reserved
15:08	IBI_ID	R	8'h00	<p>IBI Identifier</p> <p>This field indicates the Slave address byte along with the RnW bit received from the slave device.</p>
7:00	DATA_LENGTH	R	8'h00	<p>IBI Data Length</p> <p>Number of data bytes received from the Slave device either along with the SIR or with the auto-command (read). The data length is limited by QUEUE_THLD_CTRL[IBI_DATA_THLD]</p>

The controller receives the bytes in the same order as on the bus and places the first byte received in the Least Significant Byte (LSB) position, and the following bytes in the LSB + 1 location, until all the four bytes of a word are received in the IBI Queue. The Data is always preceded with IBI status and indicates the number of valid IBI data bytes as shown in [Figure 2-14](#). For more information on how IBI status and data are packed and arranged in IBI Queue, refer “[In-Band Interrupt \(IBI\) Detection and Handling](#)” on page [47](#).

Figure 2-14 IBI with Payload Data Structure

2.6.6.4 Device Address Table Data Structure

The Device Address Table is used to store the addresses of the devices that are attached to the I3C bus. The Device Address Table is available only in Master mode. It is referred by the Address Assignment Command and the Transfer Command through the DEV_INDEX field for getting the programmed target addresses.

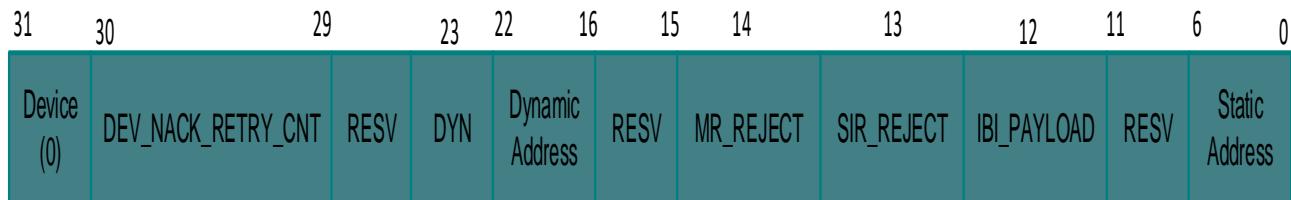
Figure 2-15 Device Address Table

There are three types of data structures that go into the Device Address Table, and these are explained as follows:

- [Figure 2-16](#) shows the structure used to represent the I3C device with Static Address. The Dynamic Address field indicates the address to be assigned for the device with the Static Address when using SETDASA command. For transfer commands, the Dynamic Address is used as the target address for all communications.

Figure 2-16 I3C Device with Static Address

- [Figure 2-17](#) shows the structure used to represent the pure I3C device with the Dynamic Address. The Dynamic Address field indicates the address to be assigned for the winning I3C device when using ENTDA command. The DYN field represents the Parity bit of 7-bit Dynamic address used to transfer along with Dynamic address during ENTDA command. For transfer commands, the Dynamic Address is used as the target address for all communications.

Figure 2-17 I3C Device with Dynamic Address

- [Figure 2-18](#) shows the structure used to represent the I2C device with the Static Address. For transfer commands, the Static Address is used as the target address for all communications. Bit[31] must be set to 1 to perform I2C protocol on the bus.

Figure 2-18 I2C Device with Static Address

2.6.6.4.1 Device Address Table Registers

The Device Address Table (DAT) is used to store the addresses of the devices that are attached to the I3C bus. The Device Address Table registers are implemented in the external RAM. To initiate transfers to a

device on the I3C bus, its information should be available in the DAT register. One DAT register is required per device. The DAT registers are explained in “[Register Descriptions](#)” on page [213](#).

The DAT registers are defined differently based on IC_HAS_IBI_DATA parameter.

- If IC_HAS_IBI_DATA=0, the DAT registers are named as DEV_ADDR_TABLE_LOCx, where x ranges from 1 to 11.
- If IC_HAS_IBI_DATA=1, the DAT registers are named as DEV_ADDR_TABLEx_LOC1, where x ranges from 1 to 11.

The register offset address specified for DAT registers in “[Register Descriptions](#)” on page [213](#) is for a specific configuration and DAT register offset addresses change based on the configuration. Refer The Device Address Table Pointer register (DEVICE_ADDR_TABLE_POINTER, offset- 0x5c) for the Start Address and Depth of Device Address Table.

2.6.6.4.2 DEV_ADDR_TABLE_LOCx

This register is used in IC_HAS_IBI_DATA=0 configuration, and the definition of DEV_ADDR_TABLE_LOC1 through DEV_ADDR_TABLE_LOC11 follows what is outlined for “[DEV_ADDR_TABLE_LOC1](#)” on page [424](#).

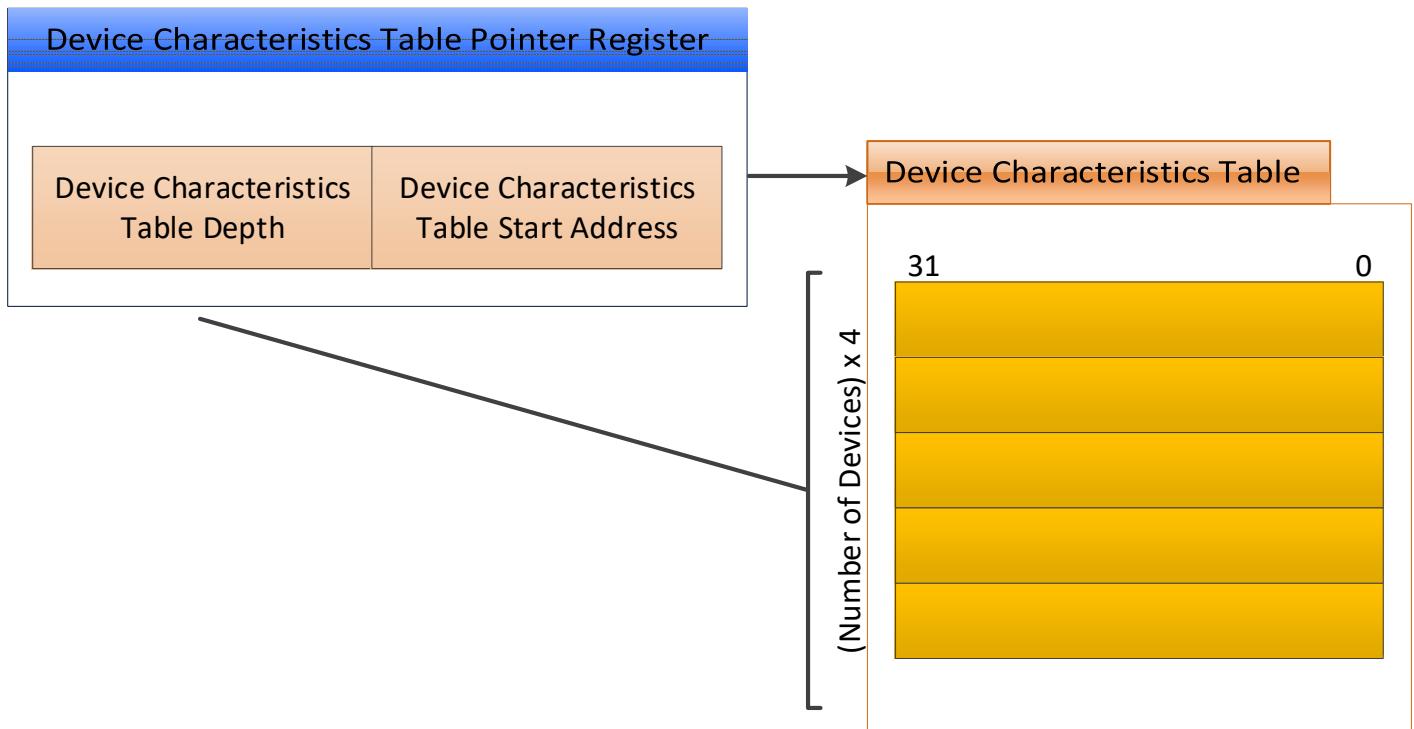
2.6.6.4.3 DEV_ADDR_TABLEx_LOC1

This register is used in IC_HAS_IBI_DATA=1 configuration and the definition of DEV_ADDR_TABLE1_LOC1 through DEV_ADDR_TABLE11_LOC1 follows what is outlined for “[DEV_ADDR_TABLE1_LOC1](#)” on page [421](#).

2.6.6.5 Device Characteristics Table Data Structure

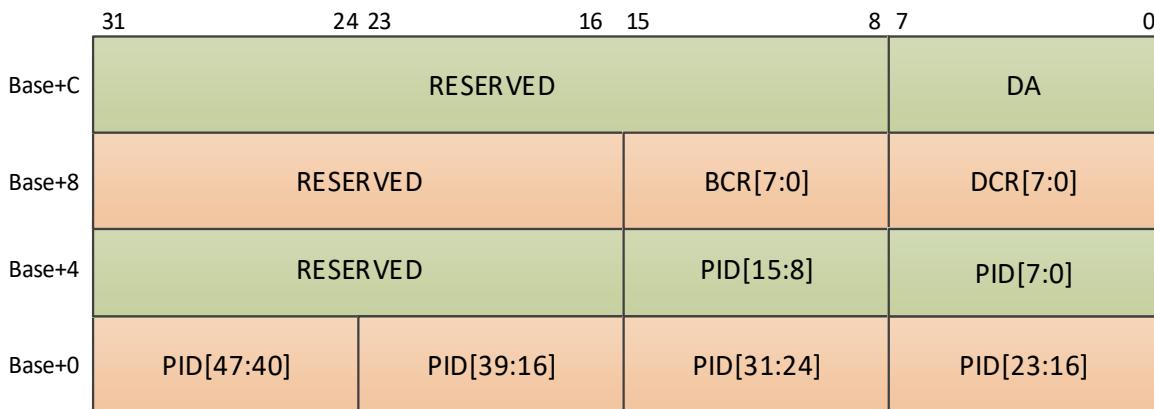
The Device Characteristics Table is used to capture the received device characteristics information (PID, BCR, DCR) and assigned dynamic address by the Master for the participating devices during ENTDAA CCC command execution (Main Master mode) and the 4 bytes (per device) received during DEFSLVS CCC command reception (Secondary Master mode).

The information captured during ENTDAA CCC can be used by the application to know what dynamic address is assigned to a particular Slave with the captured characteristics. The application should update the DAT table based on the characteristics received during the ENTDAA procedure. For example, whether or not a device is capable of sending IBI payload indicated by (BCR[2]) should reflect in ‘IBI_PAYLOAD’ bit of DAT table for that particular Slave.

Figure 2-19 Device Characteristics Table

The following two types of data structure go into the Device Characteristic Table:

- Device Characteristics Table structure during ENTDAA
- Device Characteristics Table Structure during DEFLSVS.

Figure 2-20 Device Characteristics Table Structure During ENTDAA

- [Figure 2-21](#) shows the structure used only in Non-current Master (including Secondary Master) to capture the device characteristic details during the DEFLSVS command. The data structure captures the details of the connected I3C devices which includes Dynamic Address, DCR, BCR, and the Static Address (if applicable).

Figure 2-21 Device Characteristics Table Structure During DEFSLVS Command

2.6.6.5.1 Device Characteristic Table Registers

There are two types of Device Characteristic Table (DCT): Device Characteristic Table structure for ENTDAA and Device Characteristic Table structure for DEFSLVS. These two types of registers are defined in “[Register Descriptions](#)” on page [213](#).

The DEV_CHAR_TABLEx_LOCy is used to capture characteristic information during ENTDAA and SEC_DEV_CHAR_TABLEx is used to capture characteristic information during DEFSLVS.

The DEV_CHAR_TABLEx_LOCy and SEC_DEV_CHAR_TABLEx address offsets are same and they share the same locations in External RAM. DEV_CHAR_TABLEx_LOCy should be used in Master mode of operation and SEC_DEV_CHAR_TABLEx should be used in Slave mode of operation (in Secondary Master).

2.6.6.5.2 DEV_CHAR_TABLEx_LOCy

The definition of DEV_CHAR_TABLEx_LOCy, where x ranges from 1 – 11 and y from 1 – 4 is in “[DEV_CHAR_TABLE1_LOC1](#)” on page [416](#), “[DEV_CHAR_TABLE1_LOC2](#)” on page [418](#), “[DEV_CHAR_TABLE1_LOC3](#)” on page [308](#), and “[DEV_CHAR_TABLE1_LOC4](#)” on page [420](#).

2.6.6.5.3 SEC_DEV_CHAR_TABLEx

The definition of SEC_DEV_CHAR_TABLEx, where x ranges from 1 – 11 is in “[SEC_DEV_CHAR_TABLE1](#)” on page [417](#).



Note Application writes DAT registers and the controller writes DCT registers. As SPRAM is not initialized, reading DAT and DCT registers before writing to them results in undefined values.

2.6.7 Master Data Structures in HCI Mode

This section describes the Master data structures of DWC_mipi_i3c in HCI mode. This chapter consists of following sections:

- “[Command and Response Structure](#)”
- “[IBI Status Descriptor](#)”
- “[Device Address Table \(DAT\)](#)”
- “[Device Characteristic Table \(DCT\)](#)”

2.6.7.1 Command and Response Structure

The command port of DWC_mipi_i3c accepts the following four types of command structures to initiate a transfer on the DWC_mipi_i3c bus:

- Address Assignment Command

- Immediate Data Transfer Command
- Regular Data Transfer Command
- Combo Transfer Command

Each Command comprises of two 32-bit words. These commands are used by the controller to initiate transfers on the bus. The controller issues the command on the DWC_mipi_i3c line only after receiving two 32-bit words. If the controller receives a partial command, then controller waits until it receives the complete command to initiate the transfer.

2.6.7.1.1 Address Assignment Command

This command is used for Address assignment to the Slaves through either ENTDAA Broadcast transfer or SETDASA Directed transfer. The ENTDAA transfer is used to assign the dynamic addresses to the Slaves by receiving the PID, BCR, and DCR bytes of the Slaves. The SETDASA transfer is the compact version of the ENTDAA transfer, where the host controller assigns dynamic address to the Slaves for which static address is already known. The prerequisite to issuing Address assignment command is the that host controller must program the DAT table. The host controller application must issue the Address assignment command in following cases:

- During Initialization of the DWC_mipi_i3c system.
- If any new device is joined to the system and indicated through Hot-Join IBI.

The command structure of Address Assignment command is as described in [Table 2-10](#).

Table 2-10 Address Assignment Transfer Command Structure in HCI-Mode

Bits	Field Name	Memory Access	Reset Value	Description
63:32	Reserved	-	-	-
31	TOC	W	0x0	<p>Terminate on Completion Controls which Bus condition to issue after the transfer command completes. Values:</p> <ul style="list-style-type: none"> ■ 0x0: RESTART: Repeated Start (Sr) is issued at the end of transfer. ■ 0x1: STOP: Stop (P) is issued at end of the transfer. <p>Note: TOC must always be set to 0x1 for ENTDAA, and this bit is meaningful only for SETDASA transfers.</p>
30	ROC	W	0x0	<p>Response on Completion Controls whether Response Status is sent after successful completion of the transfer command. The successful completion is read from register RESPONSE_QUEUE_PORT. Response Status is always sent after an unsuccessful transfer. Values:</p> <ul style="list-style-type: none"> ■ 0x0: NOT_REQUIRED: Response Status is not required. ■ 0x1: REQUIRED: Response Status is required.

Table 2-10 Address Assignment Transfer Command Structure in HCI-Mode (Continued)

Bits	Field Name	Memory Access	Reset Value	Description
29:26	DEV_COUNT	W	0x0	Device Count Indicates the number of devices that the address is assigned to.
25:20	RESERVED2	—	—	—
19:16	DEV_INDEX	W	0x0	Device Index Indicates DAT table index where information related to Static and Device addressing is stored. The controller starts with the information from this Index and sequentially increments to the next location for assigning the address to subsequent devices until DEV_COUNT number of times.
15	RESERVED3	—	—	—
14:7	CMD	W	0x0	Transfer Command CCC Value This field specifies CCC code indicating whether Address Assignment uses ENTDAA or SETDASA commands. The field comprises entire command code (ENTDAA or SETDASA)
6:3	TID	W	0x0	Transaction ID This field is used as tag for the command.
2:0	CMD_ATTR	W	0x0	Command Attributes This field defines Command Type and bit-field format. Values: <ul style="list-style-type: none"> ■ 0x0: XFER: Regular Transfer ■ 0x1: IMMED_DATA_XFER: Immediate data transfer ■ 0x2: ADDR_ASSGN_CMD: Address assignment command ■ 0x3: WWR_COMBO_XFER: Write + Write/Read combo transfer ■ 0x7: INTERNAL_CONTROL: Internal control command Note: Values 0x4-0x6 are reserved.



This command is not used for the address assignment commands ‘SETAASA’ Broadcast CCC transfer and ‘SETNEWDA’ Directed CCC transfer. The application must use either ‘Immediate Data Transfer’ or ‘Regular Data Transfer’ command for issuing SETAASA or SETNEWDA command.

2.6.7.1.2 Immediate Data Transfer Command

This command is used for short data write transfers including CCC transfers and Private SDR transfers(I3C/I2C), where the payload ranges from 0 to 4 bytes. The data payload is embedded inside the command itself.

The command structure of Immediate data transfer command is described in [Table 2-8](#).

Table 2-11 Immediate Transfer Command Structure in HCI-Mode

Bits	Field Name	Memory Access	Reset Value	Description
(32+31:24	DATA_BYTE_4	W	0x0	Immediate Data Transfer Data Byte 4 Direct argument
(32+23:16	DATA_BYTE_3	W	0x0	Immediate Data Transfer Data Byte 3 Direct argument
(32+15:8	DATA_BYTE_2	W	0x0	Immediate Data Transfer Data Byte 2 Direct argument
(32+7:0	DATA_BYTE_1	W	0x0	Immediate Data Transfer Data Byte 1 Direct argument
31	TOC	W	0x0	Terminate on Completion Controls which bus condition is issued after completion of the data transfer. Values: <ul style="list-style-type: none"> ■ 0x0: RESTART: Repeated Start (Sr) is issued at the end of transfer ■ 0x1: STOP: Stop (P) is issued at end of the transfer
30	ROC	W	0x0	Immediate Data Transfer Response on Completion Controls whether a Response Status is required after successful completion of the data transfer command. The successful completion is read from RESPONSE_QUEUE_PORT register. The response status is always sent upon unsuccessful transfer. Values: <ul style="list-style-type: none"> ■ 0x0: NOT_REQUIRED: Response Status is not required ■ 0x1: REQUIRED: Response Status is required
29	RNW	W	0x0	Immediate Data Transfer R/ W Identifies direction of the transfer. This field must always be set to 1'b0, because immediate transfers are valid for Write transactions only. Values: <ul style="list-style-type: none"> ■ 0x0: WRITE transfer ■ 0x1: READ transfer

Table 2-11 Immediate Transfer Command Structure in HCI-Mode (Continued)

Bits	Field Name	Memory Access	Reset Value	Description
28:26	MODE/SPEED	W	0x0	<p>Immediate Data Transfer Mode and Speed</p> <p>Mode and speed of the transfer. This field is used to program the mode and speed in which I3C or I2C are to be initiated.</p> <p>The value of this field is decoded based on mode, which can be either I3C or I2C, as determined by the 'DEVICE' field in the Device Address Table entry pointed by the 'DEV_INDEX' field of the command.</p> <p>DEVICE value of 1 indicates I2C mode. DEVICE value of 0 indicates I3C mode.</p> <p>Values (I3C mode):</p> <ul style="list-style-type: none"> ■ 0x0: I3C SDR0. I3C SDR0 represents standard SDR Speed (0 to fSCL Max). ■ 0x1: I3C SDR1. The I3C SDR1 represents Sustainable Data Rate of 8 MHz. ■ 0x2: I3C SDR2. The I3C SDR2 represents Sustainable Data Rate of 6 MHz ■ 0x3: I3C SDR3. The I3C SDR3 represents Sustainable Data Rate of 4 MHz. ■ 0x4: I3C SDR4. The I3C SDR4 represents Sustainable Data Rate of 2 MHz. ■ 0x5: Reserved (I3C HDR-TS:HDR-Ternary Mode) <p>Note: Whether TSP or TSL is used depends on global DEVICE_CTRL.I2C_SLAVE_PRESENT bit.</p> <ul style="list-style-type: none"> ■ 0x6: I3C HDR-DDR (HDR-Double Data Rate Mode) ■ 0x7: RESERVED <p>Values(I2C mode):</p> <ul style="list-style-type: none"> ■ 0x0: I2C FM ■ 0x1: I2C FM+ ■ 0x2: I2C UDR1 (I2C SS). The I2C UDR1 represents the user-defined data rate 1 (Used for standard speed mode). ■ 0x3: I2C UDR2. The I2C UDR2 represents the user-defined data rate 2. Currently, it is Reserved. ■ 0x4: I2C UDR3. The I2C UDR3 represents the user-defined data rate 3. Currently, it is Reserved. ■ 0x5 -0x7: RESERVED

Table 2-11 Immediate Transfer Command Structure in HCI-Mode (Continued)

Bits	Field Name	Memory Access	Reset Value	Description
25:23	BYTE_CNT	W	0x0	<p>Immediate Data Transfer Byte Count Number of valid data bytes to use in this Immediate Data Transfer Descriptor.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0: No payload ■ 1-4: Bytes 1-4 valid ■ 5-7: Reserved
22:20	RESERVED			
19:16	DEV_INDEX	W	0x0	<p>Immediate Data Transfer Device Index Device Index indicates DAT table index, where information related to Static and Device addressing is stored. This field indicates Slave device addressed with the transfer.</p>
15	CP	W	0x0	<p>Immediate Data Transfer Command Present Command Present. Indicates whether CMD field is valid for CCC or HDR Transfer.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: TRANFSER: Describes SDR transfer. CMD field is not valid. ■ 0x1: CCC_HDR: Describes CCC or HDR transfer. CMD field is valid.
14:7	CMD	W	0x0	<p>Immediate Data Transfer CCC / HDR Command Code Value Transfer Command field specifies Command code for CCC (8 bit) or HDR (7 bit).</p>
6:3	TID	W	0x0	<p>Immediate Data Transfer Transaction ID Used as an identification tag for this command. This field is populated by the software driver, and the same value is reflected in the Response Descriptor.</p>
2:0	CMD_ATTR	W	0x0	<p>Immediate Data Transfer Command Attribute Command attribute field defines command type and bit-field format.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: XFER: Regular Transfer ■ 0x1: IMMED_DATA_XFER: Immediate Data Transfer ■ 0x2: ADDR_ASSGN_CMD: Address Assignment Command ■ 0x3: WWR_COMBO_XFER: Write + Write/Read Combo Transfer ■ 0x7: INTERNAL_CONTROL: Internal Control command <p>Note: Values 0x4-0x6 are reserved.</p>



This command is not supported for HDR transfers because in HDR modes, at the most two words can be transferred in this format.

2.6.7.1.3 Regular Data Transfer Command

This command is used for CCC transfers, private I3C/I2C SDR data transfers, and private I3C HDR transfers. The application must transfer the Write-data payload or retrieve the Read-data payload from DATA_PORT based on the type of the command (Read/Write). You can use single regular data transfer command for the payload which can range from 0 to 65535 bytes. If the payload is more than 65535 bytes, the application must initiate multiple regular data transfer commands based on the required data length of the transfer.

The command structure of the Regular Data Transfer command is described in [Table 2-12](#).

Table 2-12 Regular Transfer Command Structure in HCI-Mode

Bits	Field Name	Memory Access	Reset Value	Description
(32+):31:16	DATA_LENGTH	W	0x0	<p>Data Transfer Data Length Data Length field is used to indicate the length of the transfer in bytes.</p> <p>Note: This field must be set to a non-zero value, except for CCCs that does not have a defined payload.</p>
(32+):15:0	Reserved	—	—	—
31	TOC	W	0x0	<p>Data Transfer Terminate on Completion Controls which bus condition is issued after completion of the transfer.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: RESTART: Repeated Start (Sr) is issued at the end of transfer ■ 0x1: STOP: Stop (P) is issued at end of the transfer
30	ROC	W	0x0	<p>Data Transfer Response on Completion Controls whether Response Status is required after successful completion of the transfer command. The successful completion is read from RESPONSE_QUEUE_PORT register. The Response Status is always sent after an unsuccessful transfer.</p> <p>Note: ROC value of 0 is only valid for requests in PIO mode. In DMA mode, you must always write '1' to populate Response Descriptor.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: NOT_REQUIRED: Response Status is not required ■ 0x1: REQUIRED: Response Status is required

Table 2-12 Regular Transfer Command Structure in HCI-Mode (Continued)

Bits	Field Name	Memory Access	Reset Value	Description
29	RNW	W	0x0	<p>Data Transfer R/ W Identifies the direction of this transfer.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: WRITE: Write transfer ■ 0x1: READ: Read transfer
28:26	MODE/SPEED	W	0x0	<p>Data Transfer Speed and Mode Speed and mode of the transfer. This field is used to program the speed/mode in which I3C or I2C must be initiated by the controller.</p> <p>The value of this field is decoded based on whether the transfer is targeted to I3C device or I2C device as determined by the 'DEVICE' field in the Device address table pointed by the 'DEV_INDEX' field of the command.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: I3C SDR0/ I2C Fm: Note: I3C SDR0 represents standard SDR Speed (0 to fSCL Max) ■ 0x1: I3C SDR1 / I2C Fm+ Note: I3C SDR1 represents Sustainable data rate of 8 MHz ■ 0x2: I3C SDR2 / I2C UD (I2C SS) Note: I3C SDR2 represents Sustainable data rate of 6 MHz. I2C UD represents the user-defined standard speed I2C Speed. ■ 0x3: I3C SDR3/ I2C Reserved Note: I3C SDR3 represents sustainable data rate of 4 MHz. I2C UD represents the user-defined I2C Speed (currently it is Reserved). ■ 0x4: I3C SDR4/ I2C Reserved Note: I3C SDR4 represents sustainable data date of 2 MHz. I2C UD represents the user-defined I2C Speed (Currently it is Reserved). ■ 0x5: Reserved (I3C HDR-TS: HDR-Ternary Mode) / I2C Reserved Note: <ol style="list-style-type: none"> 1. Whether TSP or TSL is used depends on global DEVICE_CTRL.I2C_SLAVE_PRESENT bit. ■ 0x6: I3C HDR-DDR (HDR-Double Data Rate Mode) / I2C Reserved ■ 0x7: RESERVED
25:20	RESERVED	-	-	-

Table 2-12 Regular Transfer Command Structure in HCI-Mode (Continued)

Bits	Field Name	Memory Access	Reset Value	Description
19:16	DEV_INDEX	W	0x0	Data Transfer Device Index Device Index indicates DAT table index, where information related to static and device addressing is stored. Field indicates Slave device addressed with the transfer.
15	CP	W	0x0	Data Transfer Command Present Command present. Indicates whether CMD field is valid for CCC or HDR Transfer. Values: <ul style="list-style-type: none"> ■ 0x0: TRANSFER: Describes SDR transfer. CMD field is not valid. ■ 0x1: CCC_HDR: Describes CCC or HDR transfer. CMD field is valid.
14:7	CMD	W	0x0	Data Transfer CCC / HDR Command Code Value Transfer Command field specifies command code for CCC (8 bit) or HDR (7 bit).
6:3	TID	W	0x0	Data Transfer Transaction ID Transaction ID field is used as identification tag for the command.
2:0	CMD_ATTR	W	0x0	Data Transfer Command Attribute Command Attribute field defines Command Type and Bit-Field Format. Values: <ul style="list-style-type: none"> ■ 0x0: XFER: Regular Transfer ■ 0x1: IMMED_DATA_XFER: Immediate Data Transfer ■ 0x2: ADDR_ASSGN_CMD: Address Assignment Command ■ 0x3: WWR_COMBO_XFER: Write + Write/Read Combo transfer ■ 0x7: INTERNAL_CONTROL: Internal Control command Note: Values 0x4-0x6 are reserved.

2.6.7.1.4 Combo Transfer Command

This command is used to issue the ‘Write followed by Write transfer’ or ‘Write followed by Read transfer’ as a single atomic transaction. This command is useful where the application has to indicate the sub-address of the device first and then either write or read data from the sub-address pointed by the first write transfer. The first write transfer payload is embedded inside the command itself and the subsequent transfer Write-

data/Read-data is written or accessed through DATA_PORT. The Combo transfer is applicable to both I3C and I2C transfers. The command structure of the combo transfer command is described in [Table 2-13](#).

Table 2-13 **Combo Transfer Command Structure in HCI-Mode**

Bits	Field Name	Memory Access	Reset Value	Description
(32+31:16)	DATA_LENGTH	W	0x0	Combo Transfer Data Length Data length field is used to indicate the length of the transfer in bytes. Note: This field must be set to a non-zero value.
(32+15:0)	OFFSET/SUB OFFSET	-	-	Combo Transfer Offset / Sub-Offset Offset/sub-offset to select offset of target operation
31	TOC	W	0x0	Combo Transfer Terminate on Completion Controls which bus condition is issued after completion of the transfer. Values: <ul style="list-style-type: none"> ■ 0x0: RESTART: Repeated Start (Sr) is issued at the end of transfer ■ 0x1: STOP: Stop (P) is issued at end of the transfer
30	ROC	W	0x0	Combo Transfer Response on Completion Response on completion controls whether Response Status is required after successful completion of the transfer command. The successful completion is read from RESPONSE_QUEUE_PORT register. Response status is always sent upon unsuccessful transfer. Values: <ul style="list-style-type: none"> ■ 0x0: NOT_REQUIRED: Response Status is not required ■ 0x1: REQUIRED: Response Status is required
29	RNW	W	0x0	Combo Transfer R/ W This field identifies direction of the transfer. Values: <ul style="list-style-type: none"> ■ 0x0: WRITE: Write transfer ■ 0x1: READ: Read transfer

Table 2-13 Combo Transfer Command Structure in HCI-Mode (Continued)

Bits	Field Name	Memory Access	Reset Value	Description
28:26	MODE/SPEED	W	0x0	<p>Combo Transfer Speed and Mode</p> <p>This field is used to program the speed/mode in which I3C or I2C are to be initiated by the controller.</p> <p>The value of this field is decoded based on whether the transfer is targeted to a I3C device or I2C device as determined by the 'DEVICE' field in the Device address table pointed by the 'DEV_INDEX' field of the command.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: I3C SDR0/ I2C Fm: Note: I3C SDR0 represents standard SDR speed (0 to fSCL Max) ■ 0x1: I3C SDR1 / I2C Fm+: Note: I3C SDR1 represents sustainable data rate of 8 MHz ■ 0x2: I3C SDR2 / I2C UD (I2C SS): Note: I3C SDR2 represents sustainable data rate of 6 MHz. I2C UD represents the user-defined I2C speed (Used for Standard Speed). ■ 0x3: I3C SDR3/ I2C UD(Reserved) Note: I3C SDR3 represents sustainable data rate of 4 MHz. I2C UD represents the user-defined I2C speed (currently It is Reserved for I2C Mode). ■ 0x4: I3C SDR4/ I2C UD(Reserved) Note: I3C SDR4 represents sustainable data rate of 2 MHz. I2C UD represents the user-defined I2C speed (currently It is Reserved for I2C Mode). ■ 0x5: Reserved (I3C HDR-TS: HDR-Ternary Mode) / I2C Reserved Note: Combo transfers are supported only in SDR Mode and not in HDR Mode. ■ 0x6: Reserved (I3C HDR-DDR: HDR-Double Data Rate Mode) / I2C Reserved Note: Combo transfers are supported only in SDR Mode and not in HDR Mode. ■ 0x7: RESERVED
25	16_BIT_SUBO FFSET	W	0x0	<p>Combo Transfer Sub Offset Size</p> <p>Indicates whether the Sub-Offset is 8 or16 bits long.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: 8_BIT_SUBOFFSET: Sub-offset is 8 bits long. Value is encoded in lower byte of OFFSET / SUBOFFSET field. ■ 1'b1: 16_BIT_SUBOFFSET: Sub-offset is 16 bits long

Table 2-13 Combo Transfer Command Structure in HCI-Mode (Continued)

Bits	Field Name	Memory Access	Reset Value	Description
24	FIRST_PHASE_MODE	W	0x0	<p>Combo Transfer First Phase Mode Indicates whether the first phase of the combo transfer is executed in SDR Mode, or the mode indicated by the MODE field.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: SDR: First phase is executed in SDR mode ■ 1'b1: MODE: First phase is executed in the mode indicated by the MODE field <p>Note: This field is not supported by DWC_mipi_i3c controller, and must be set to 0.</p>
23:22	DATA_LENGTH_POSITION	W	0x0	<p>Data Length Field Position Indicates whether and where to put data length (DATA_LENGTH) in the first phase of the transfer. This field is only applicable if first phase of the transfer is executed in HDR mode.</p> <p>Whether 8-bit or 16-bit of data length field is used is indicated with 16_BIT_SUBOFFSET field. For 8-bit value, it is encoded in lower byte of DATA_LENGTH field.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 2'b0: NO: Do not put Length field ■ 2'b1: FIRST: Put Length as first field ■ 2'b2: SECOND: Put Length as second field ■ 2'b3: RESERVED: Do not use <p>Note: This field is not supported by DWC_mipi_i3c controller and must be set to 0.</p>
24:20	RESERVED	—	—	—
19:16	DEV_INDEX	W	0x0	<p>Combo Transfer Device Index Device Index indicates DAT table index where information related to static and device addressing is stored. Field indicates Slave Device addressed with the transfer.</p>
15	CP	W	0x0	<p>Combo Transfer Command Present Indicates whether the CMD field is valid for a HDR Transfer.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b0: TRANSFER: This structure describes a SDR transfer, so the CMD field is not valid. ■ 1'b1: CCC_HDR: This structure describes a HDR transfer, so the CMD field is valid. <p>Note: This field is not supported by DWC_mipi_i3c controller and must be set to 0.</p>

Table 2-13 Combo Transfer Command Structure in HCI-Mode (Continued)

Bits	Field Name	Memory Access	Reset Value	Description
14:7	CMD	W	0x0	<p>Combo Transfer HDR Command Code Value Specifies the I3C command code (7 bits). Note: This field is not supported by DWC_mipi_i3c controller and must be set to 0.</p>
15:7	Reserved	—	—	—
6:3	TID	W	0x0	<p>Combo Transfer Transaction ID Transaction ID field is used as an identification tag for the command.</p>
2:0	CMD_ATTR	W	0x0	<p>Combo Transfer Command Attribute Command attribute field defines Command type and bit-field format. Values:</p> <ul style="list-style-type: none"> ■ 0x0: XFER: Regular Transfer ■ 0x1: IMMED_DATA_XFER: Immediate Data Transfer ■ 0x2: ADDR_ASSGN_CMD: Address Assignment Command ■ 0x3: WWR_COMBO_XFER: Write + Write/Read Combo Transfer ■ 0x7: INTERNAL_CONTROL: Internal Control command <p>Note: Values 0x4-0x6 are reserved.</p>



The combo transfer command supports transfers only in SDR mode.

2.6.7.1.5 Response Data Structure

The DWC_mipi_i3c controller returns the status of the transfer command into the response queue with this data structure.

Table 2-14 Response Data Structure in HCI-Mode

SL No.	Field Name	Bit-Field	Memory Access	Reset Value	Description
1	DATA_LENGTH	15:0	R	0x0	Data Length / Device Count For write transfers, this field represents the remaining data length of the transfer if the transfer is terminated early (remaining data length = requested data length - transferred data length) For read transfers, this field represents the actual amount of data received in bytes. For Address assignment command, this field represents the remaining device count.
2	Reserved	23:16	-	-	-
3	TID	27:24	R	0x0	Command/Response Transaction ID Transaction ID field is used as an identification tag for the command. This value matches one of commands sent on the bus. Note: Valid IDs are 0x0 - 0x7. Values 0x8 - 0xF are reserved.

Table 2-14 Response Data Structure in HCI-Mode (Continued)

SL No.	Field Name	Bit-Field	Memory Access	Reset Value	Description
4	ERR_STATUS	31:28	R	0x0	<p>Response Error Status Indicates the Response status, that is, either success or the error type encountered for the processed command.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: SUCCESS: Transfer successful, no error ■ 0x1: CRC: CRC Error ■ 0x2: PARITY: Parity Error ■ 0x3: FRAME: Frame Error ■ 0x4: ADDR_HEADER: Address Header Error ■ 0x5: NACK: Address NACK'ed or Dynamic Address Assignment NACK'ed ■ 0x6: OVL: Receive Overflow or Transfer Underflow Error ■ 0x7: RESERVED ■ 0x8: ABORTED: Aborted ■ 0x9: I2C_WR_DATA_NACK: NACK received for the I2C Write Data transfer ■ 0xA: NOT_SUPPORTED: Command with specific parameters not supported by the Host Controller implementation (For example, specific Internal control codes may not be supported) ■ 0xB – 0xF: RESERVED

2.6.7.2 IBI Status Descriptor

The IBI status descriptor is a read-only structure describing an IBI event received from a Slave device on the DWC_mipi_i3c bus.

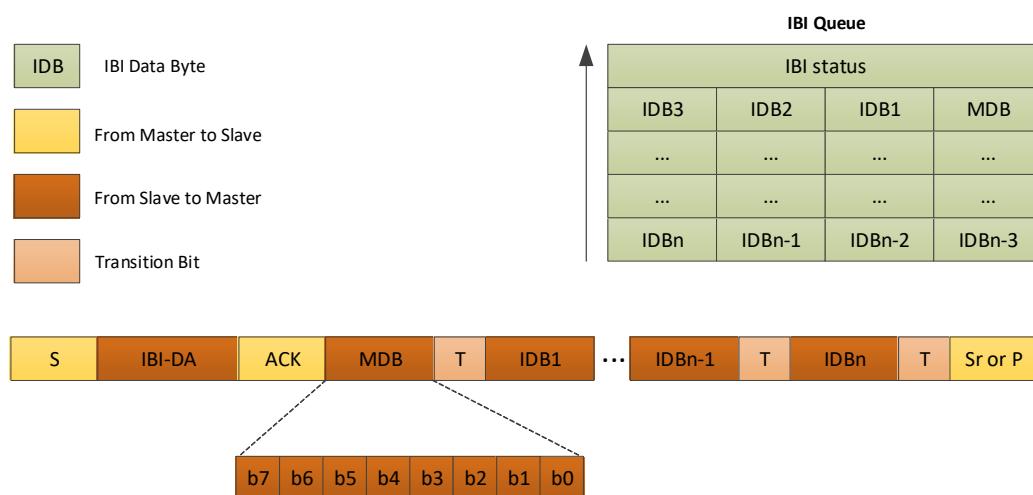
Table 2-15 IBI Status Data Structure in HCI-Mode

Bits	Field Name	Access Type	Reset Value	Description
31	IBI_STS	R	0x0	<p>IBI Status</p> <p>Indicates the status of the response returned for the received IBI.</p> <p>1'b0: The received IBI is responded with ACK. Any non-zero value of the DATA_LEN field indicates the presence of data payload for the ACK'ed IBI.</p> <p>1'b1: The received IBI is responded with NACK. An auto disable CCC command is issued if the received IBI address is valid and matching with the DAT entry.</p> <p>If an IBI is received from an unknown address (not a valid entry in DAT), the IBI_STS is set to 1.</p>
30	ERROR	R	0x0	<p>Error</p> <p>Indicates that during IBI Auto command, an error is encountered and the data from the Slave device is partially or fully discarded.</p> <p>The following errors are detected:</p> <ul style="list-style-type: none"> ■ CRC/Parity Error (applicable for HDR transaction) ■ Slave Address NACK ■ 0x7E Address NACK ■ IBI buffer overflow (applicable for IBI HDR modes) <p>1'b0: No Error. Transaction completed with no errors.</p> <p>1'b1: Error. Error encountered during IBI Auto command phase.</p>
29	RESV	R	0x0	-
28:26	HW_CONTEXT	R	0x0	<p>IBI Hardware Context</p> <p>Hardware specific context for IBI processing. Opaque for driver.</p> <p>Note: This field is reserved for DWC_mipi_i3c controller.</p>
25	TS	R	0x0	<p>IBI Timestamp Present</p> <p>Indicates whether a timestamp is available for the IBI.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 1'b1: ON: IBI is timestamped ■ 1'b0: OFF: IBI is not timestamped <p>Note: This field is reserved for DWC_mipi_i3c controller.</p>
24	LAST_STATUS	R	0x0	<p>Last Status</p> <p>When set, indicates that this status is the last for the received IBI.</p> <p>If the payload of the received SIR exceeds the programmed IBI data threshold, then the controller splits the IBI payload into multiple chunks of IBI_DATA_THLD size (max) which includes the timestamp bytes, if enabled.</p>
23:16	RESV	R	8'h00	Reserved

Table 2-15 IBI Status Data Structure in HCI-Mode (Continued)

Bits	Field Name	Access Type	Reset Value	Description
15:8	IBI_ID	R	8'h00	IBI Identifier This field indicates the Slave address byte along with the R/W bit received from the Slave device.
7:0	DATA_LENGTH	R	8'h00	IBI Data Length Number of data bytes received from the Slave device either along with the SIR or with the auto-command (read). The data length is limited by QUEUE_THLD_CTRL[IBI_DATA_THLD]

The controller receives the bytes in the same order as on the bus and places the first byte received in the Least Significant Byte (LSB) position, and the following bytes in the LSB + 1 location, until all the four bytes of a word are received in the IBI Queue. The Data is always preceded with IBI status and indicates the number of valid IBI data bytes as shown in [Figure 2-22](#). Refer “[In-Band Interrupt \(IBI\) Detection and Handling](#)” on page [47](#) for more information on how IBI status and data are packed and arranged in IBI Queue.

Figure 2-22 IBI with payload Data structure

2.6.7.3 Device Address Table (DAT)

The Device Address Table stores the information of the Device addresses and attributes (I2C or I3C, IBI Capable, and so on) of the devices (slaves) present in the DWC_mipi_i3c bus. Each entity is composed of two 32-bit DW. These entries are referred through ‘Device Index’ field of the command. Device Address Table Offset, in Capabilities registers, indicates offset to device Address Table and number of entries. There are two entries for each device and is described with Device Address Table structure in [Table 2-16](#). The DAT table is considered as part of the register set. Although DAT structure is presented and also treated just as any other software programmable register, DAT is implemented in SPRAM. This implementation detail is transparent to the software and the use model (each location of the DAT (present in the SPRAM) can be access by a register offset).

Table 2-16 Device Address Table Structure in HCI-Mode

Bits	Field Name	Memory Access	Reset Value	Description
(32+):31:27	Reserved	—	—	—
(32+):26:19	AUTOCMD_HDR_CODE	R/W	0x0	<p>Device Auto-Command HDR Command Code Specifies Auto-Command Read Command Code. Valid only if a Read is executed in HDR Mode. If the value is below 0x80, then the Host Controller uses 0x80 for HDR Command Code.</p>
(32+):18:16	AUTOCMD_MODE	R/W	0x0	<p>Device Auto-Command Mode Speed and mode of the IBI Auto command transfer. This field is used to program the speed/mode in which IBI Auto command transfer must be initiated by the controller.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: I3C SDR0: Note: I3C SDR0 represents standard SDR Speed (0 to fSCL Max) ■ 0x1: I3C SDR1 Note: I3C SDR1 represents sustainable data rate of 8 MHz ■ 0x2: I3C SDR2 Note: I3C SDR2 represents sustainable data rate of 6 MHz. ■ 0x3: I3C SDR3 Note: I3C SDR3 represents sustainable data rate of 4 MHz. ■ 0x4: I3C SDR4 Note: I3C SDR4 represents sustainable data rate of 2 MHz. ■ 0x5: Reserved (I3C HDR-TS: HDR-Ternary Mode) / I2C Reserved Note: IBI Auto command Transfer are supported in only SDR Mode and not in HDR Mode. ■ 0x6: Reserved (I3C HDR-DDR: HDR-Double Data Rate Mode) / I2C Reserved Note: IBI Auto command Transfer are supported in only SDR Mode and not in HDR Mode. ■ 0x7: RESERVED

Table 2-16 Device Address Table Structure in HCI-Mode (Continued)

Bits	Field Name	Memory Access	Reset Value	Description
(32+15:8	AUTOCMD VALUE	R/W	0x0	Device Auto-Command IBI Mandatory Byte Value Value of IBI mandatory byte that triggers automatic Read transaction on the bus (Auto Command feature)
(32+7:0	AUTOCMD_MASK	R/W	0x0	Device Auto-Command Mask Mask of IBI mandatory byte that triggers an automatic Read transaction on the bus (Auto Command feature)
31	DEVICE	R/W	0x0	Device Type Type of Device Values: <ul style="list-style-type: none"> ■ 0x0: I3C: I3C Device ■ 0x1: I2C: I2C Device
30:29	DEV_NACK_RETRY_CNT	R/W	0x0	Device NACK Retry Count Device specific retry count
28:26	RING_ID	R/W	0x0	Device Ring Group ID Ring group identification. This field is used to put IBI from specific device to appropriate ring bundle. Note: This field is reserved for DWC_mipi_i3c controller.
25:24	Reserved	R/W	0x0	—
23:16	DYNAMIC_ADDRESS	R/W	0x0	Device I3C Dynamic Address I3C Dynamic Address (with parity bit) Parity bit is encoded in bit 23.
15	TS	R/W	0x0	Device IBI Timestamp Marker for Timestamping IBI for specific device. Whether IBI was timestamped or not is provided in IBI Status Descriptor Values: <ul style="list-style-type: none"> ■ 0x0: NO_TS: Do not timestamp IBI with Master Timestamps ■ 0x1: TS: Timestamp IBI with Master Timestamps Note: This field is reserved for DWC_mipi_i3c controller.

Table 2-16 Device Address Table Structure in HCI-Mode (Continued)

Bits	Field Name	Memory Access	Reset Value	Description
14	MR_REJECT	R/W	0x0	<p>Device In-Band Master Request Reject In-Band Master Request Reject field is used to control, per Device, whether to accept Master request from devices. This bit is only valid if the host controller declares non-current Master capability.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: ACCEPT: ACK the Master Request ■ 0x1: REJECT: NACK the Master Request and send auto disable CCC
13	SIR_REJECT	R/W	0x0	<p>Device In-Band Slave Interrupt Request Reject In-Band Slave interrupt request reject field is used to control, per device, whether to accept Slave Interrupt request from devices.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: ACCEPT: ACK the SIR ■ 0x1: REJECT: NACK the SIR and send auto disable CCC
12	IBI_PAYLOAD	R/W	0x0	<p>Device IBI Payload Data payload. This field reflects the IBI payload bit in the Device's Bus Characteristics Register (BCR). During IBI handling for this device, the Master uses this field to determine whether or not to drive reception of the IBI data payload. Data continuation is indicated by the T-Bit.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0: NO_PAYLOAD: IBI does not carry data payload ■ 0x1: PAYLOAD: IBI carries data payload
11:7	Reserved	R/W	0x0	—
6:0	STATIC_ADDRESS	R/W	0x0	I3C/I2C Static Address

Table 2-16 is populated by application and controller uses it in following cases:

- Dynamic Address Assignment process: Application control over DA allows for device prioritization in terms of address arbitration for IBIs from Slaves.
- Regular Data transfers: Controller retrieves the dynamic address, I2C or I3C Device, Device NACK retry count fields which are required to initiate the transfer on the line.

- SIR IBI transfers: Controller retrieves the dynamic address, IBI related fields like Auto-command Mask, which are required for the controller to decide the next steps to perform based on the incoming IBI and MDB.

2.6.7.4 Device Characteristic Table (DCT)

Device Characteristic Table captures the device characteristics (PID, BCR, DCR) and the assigned dynamic address of the participating services during ENTDAA procedure. Each entity consists of four '32-DW' locations. This information can be used by the application to know what dynamic address is assigned to a particular Slave with the captured characteristics.

The application should update the DAT table based on the characteristics received during the ENTDAA procedure, like whether the device is capable of sending IBI payload or not (BCR[2]), and should reflect the same in 'IBI_PAYLOAD' bit of DAT table for the particular slave.

The DCT table is considered a part of Register set, that is, it is implemented with use of regular registers as defined in [Table 2-17](#).

Device Characteristic Table offset, in Capabilities registers, indicates offset to Device Characteristic Table and number of entries. There is an entry for each device and is described in [Table 2-17](#).

Table 2-17 Device Characteristic Table

Bits	Field Name	Memory Access	Reset Value	Description
(96+)31:8	Reserved	—	—	—
(96+)7:0	DYNAMIC_ADDRESS	R	0x0	I3C Dynamic Address (with parity bit)
(64+)31:16	Reserved	—	—	—
(64+)15:8	BCR	R	0x0	Bus Characteristic Register
(64+)7:0	DCR	R	0x0	Device Characteristic Register
(32+)31:16	Reserved	—	—	—
(32+)15:0	PID	R	0x0	Provisional ID [15:0]
31:00	PID	R	0x0	Provisional ID [48:16]



Note Application writes DAT registers and the controller writes DCT registers. As SPRAM is not initialized, reading DAT and DCT registers before writing to them results in undefined values.

2.6.8 Operation Modes of DWC_mipi_i3c

2.6.8.1 Bus Communication Protocols in Non-HCI Modes

The DWC_mipi_i3c supports the following bus communication protocols in non-HCI modes to transfer the data between master and slave.

- Single Data Rate (SDR) Transfers

- High Data Rate -Double Data Rate (HDR-DDR)
- High Data Rate – Ternary Mode (HDR-TSP and HDR-TSL)

2.6.8.1.1 Single Data Rate (SDR) Transfers in Master Mode

In SDR mode, Single bit is sent on each SCL clock. All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer. After the master sends the address and R/W bit, the slave-receiver must respond to the acknowledge signal (ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high, so that the master can abort the transfer.

The single data rate transfers are initiated by the application through the command as explained in “[Command Data Structure](#)” on page 55. Based on the command initiated by the application and Device address pointed by the DEV_INDX field initiates command, the DWC_mipi_i3c controller initiate the transfers on the bus. [Table 2-18](#) illustrates the decoded transfer type based on Transfer command and Device Address Table.

Table 2-18 Transfer Type Decoding Based on Command and Device Address Table

Transfer Command (Command Port)				Device Address Table	
CP	CMD[14]	SPEED	RnW	DEVICE	Decoded Transfer Type
0	NA	0 to 4	0	0	I3C Private Write Transfer
0	NA	0 to 4	1	0	I3C Private Read Transfer
1	0	0 to 4	0	0	I3C Broadcast CCC Write Transfer
1	1	0 to 4	0	0	I3C Directed CCC Write Transfer
1	1	0 to 4	1	0	I3C Directed CCC Read Transfer
0	x	0 to 1	0	1	I2C Private Write Transfer
0	x	0 to 1	1	1	I2C Private Read Transfer
Others					Illegal Combination

2.6.8.1.2 Broadcast or Directed CCC Transfers

The Common Command Code transfers are used for the bus management procedures like

- Receiving the device characteristics
- Setting the max write or read data length
- Enabling or disabling the event generation in the I3C devices
- Informing the bus characteristics to all the existing devices.
- Entering the high data rate transfer modes

2.6.8.1.3 Broadcast CCC Write Transfers

The Broadcast CCC Write transfers are initiated on the bus based on the COMMAND_QUEUE_PORT settings as shown in [Table 2-19](#).

Table 2-19 Broadcast CCC Write Transfer Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	1	Indicates the controller to consider the CMD field.
	CMD[14]	0	Indicates the controller that the transfer is Broadcast CCC transfer.
	CMD[13:7]	0x0 – 0x7F	Indicates the CCC Command to be transferred.
	DEV_INDX	NA	This field is not used since the transfer is Broadcast CCC.
	SPEED	0 (SDR0) or 7 (I2C FM)	Indicates the controller that the transfer should go in SDR0 Speed/Mode or I2C FM Speed. Note: All CCC transfers are initiated with SDR0 Speed in I3C bus topology. CCC transfers can be either initiated with SDR0 or I2C FM speed based on the context of the bus in JEDEC bus topology.
	DBP	0 or 1	Indicates whether defining byte is present for Broadcast CCC transfer or not. <ul style="list-style-type: none"> ■ 0: Defining byte is not present ■ 1: Defining byte is present
	SDAP	0 or 1	<ul style="list-style-type: none"> ■ 0: Indicates to consider the transmit data from the Transmit FIFO if Rnw is set to 0. ■ 1: Indicates the controller to consider the transmit data from the command if RnW is set to 0.
	RnW	0	Indicates the transfer is either write or read transfer. <ul style="list-style-type: none"> ■ 0: Write Transfer ■ 1: Read Transfer
	PEC	0 or 1	Indicates whether Packet Error Check is enabled for Broadcast CCC transfer. <ul style="list-style-type: none"> ■ 0: PEC check is disabled. ■ 1: PEC check is enabled.
Transfer Argument (SDAP=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Short Data Argument (SDAP=1)	BYTE_STRB	0, 1, 3,7	Indicates the respective data bytes of Immediate command are valid.

The data in Tx-FIFO (Transfer Argument) or Data Bytes (Short Data Argument) are not required for some broadcast CCC's which does not have payload data Indicated through Data Length (Transfer Argument) or BYTE_STRB (Short Data Argument). If the Broadcast CCC does not consist of payload, then you must indicate it with zero in either Data Length or BYTE_STRB fields based on the command issued.

The DWC_mipi_i3c controller halts in case of receiving a NACK (It means no I3C device on the bus) for the address header of the Broadcast CCC transfer. The controller updates the 'ERR_STS' field with appropriate error information in the Response status, halts the controller and gives back the control to the application to resume the operation of the controller through writing '1' to the RESUME bit of the DEVICE_CTRL register after taking suitable action.

2.6.8.1.4 Directed Write and Read Transfers

The Directed CCC Write/Read transfers are initiated on the bus based on the COMMAND_QUEUE_PORT settings shown in [Table 2-20](#).

Table 2-20 Directed CCC Write/Read Transfers Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	1	Indicates the controller to consider the CMD field.
	CMD[14]	1	Indicates the controller that the transfer is Directed CCC transfer.
	CMD[13:7]	0x0 - 0x7F	Indicates the CCC Command to be transferred.
	DEV_INDX	DEV_INDX	Indicates the Index of the Device Table which consists of the slave address to be targeted.
	SPEED	0 (SDR0) or 7 (I2C FM)	Indicates the controller that the transfer should go in SDR0 Speed/Mode or I2C FM Speed. Note: All CCC transfers are initiated with SDR0 Speed in I3C bus topology. CCC transfers can be either initiated with SDR0 or I2C FM Speed based on the context of the bus in JEDEC bus topology.
	DBP	0 or 1	Indicates whether Defining Byte is present for Directed CCC transfer or not. <ul style="list-style-type: none"> ■ 0: Defining Byte is not present ■ 1: Defining Byte is present.
	SDAP	0 or 1	<ul style="list-style-type: none"> ■ 0: Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0. ■ 1: Indicates the controller to consider the transmit data from the command if RnW is set to 0.
	RnW	0 or 1	<ul style="list-style-type: none"> ■ 0: Indicates the transfer is write transfer. ■ 1: Indicates the transfer is read transfer.
	PEC	0 or 1	Indicates whether Packet Error Check is Enabled for Directed CCC transfer. <ul style="list-style-type: none"> ■ 0: PEC check is disabled. ■ 1: PEC check is enabled.
Transfer Argument (SDAP=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Short Data Argument (SDAP=1)	BYTE_STRB	0,1,3,7	Indicates the respective data bytes of the Immediate command are valid.

The Data in Tx-FIFO (Transfer Argument) or Data Bytes (Immediate Data Transfer) are not required for some directed write CCC's which does not have payload data indicated through Data Length (Transfer Argument) or BYTE_STRB (Short Data Argument). If the Directed CCC does not consist of payload, you must indicate it with zero in either Data Length or BYTE_STRB fields based on the command issued.

2.6.8.1.5 Directed CCC Transfer Targeted to Multiple Slaves

Each transfer command initiates directed CCC transfer to only one slave since it consists of only one DEV_INDX field. If the requirement is to transfer the directed CCC command to multiple devices, then you must pipeline the multiple transfer commands in the COMMAND_QUEUE_PORT with TOC bits set to zero and with the different DEV_INDX fields pointing to multiple slaves. The DWC_mipi_i3c controller decodes the pipelined 'Transfer command' during the transfer of directed CCC transfer and decides the next transfer based on the following:

- If the current command and the pipelined command have same Directed CCC, then the controller targets the next slave without ending the CCC command.
- If the current command and the pipelined command are not the same Directed CCC, then the controller ends the CCC command and starts issuing the next transfer as indicated by the pipelined transfer command.

The application can set the ROC bit to '0' for the subsequent directed CCC commands and enable the ROC bit in the last CCC command if the directed CCC transfer is targeted to multiple devices to avoid the unnecessary responses.

The DWC_mipi_i3c controller halts in case of the following conditions:

- Receiving NACK for the Address header of the directed CCC transfer (It means no I3C device on the bus).
- Receiving NACK for the Slave address of the directed CCC transfer.

The controller updates the 'ERR_STS' field with appropriate error information in the Response status and halts the controller and gives back the control to the application to resume the operation of the controller through writing '1' to the RESUME bit of the 'DEVICE_CTRL' register.

2.6.8.1.6 I3C Private Write or Read Transfers

The I3C private write and read transfers are initiated on the bus based on the COMMAND_QUEUE_PORT and settings shown in [Table 2-21](#).

Table 2-21 I3C Private Write/Read Transfers Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	0	Indicates the controller to not consider the CMD field.
	CMD[14]	NA	This field is not applicable since CP bit is set to 0
	CMD[13:7]	NA	This field is not applicable since CP bit is set to 0
	DEV_INDX	DEV_INDX	Indicates the Index of the Device Table which consists of the slave address to be targeted.
	SPEED	0 to 4	Indicates the controller that the transfer should go in SDR mode.
	SDAP	0 or 1	<ul style="list-style-type: none"> ■ 0: Indicates to consider the transmit data from the Transmit FIFO if Rnw is set to 0. ■ 1: Indicates the controller to consider the transmit data from the command if RnW is set to 0.
	RnW (Read and Write)	0 or 1	<ul style="list-style-type: none"> ■ 0: Indicates the transfer is write transfer. ■ 1: Indicates the transfer is read transfer.
	PEC	0 or 1	<ul style="list-style-type: none"> Indicates whether Packet Error Check is Enabled for Private SDR transfers. ■ 0: PEC check is disabled. ■ 1: PEC check is enabled.
Transfer Argument (CMD_ATTR=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Short Data Argument (CMD_ATTR=1)	BYTE_STRB	0,1,3,7	Indicates the respective data bytes of the Immediate command are valid.


Note

- The LEGACY_I2C_DEVICE bit in the Device Address Table pointed by the 'DEV_INDX' field of the Transfer Command should set to zero for the I3C private transfers.
- To avoid the initial latencies of the transfer, the controller uses 'TX/RX_START_THLD' before initiating the transfer. 'TX_START_THLD' ensures the threshold level of data is present in the Transmit buffer for write transfer and 'RX_START_THLD' level of space is available in the Receive buffer for the Read transfer before initiating the transfer. This threshold is only applicable for the transfers which are initiated with the START condition and not applicable for the transfers which are initiating with the RESTART condition for SDR Transfers.
- To allow the priority for IBI from the devices, the DWC_mipi_i3c controller can include Address header for the I3C private transfers by enabling the 'IBA_INCLUDE' bit in the 'DEVICE_CTRL' register since IBI always wins when arbitrated with Address header.

The DWC_mipi_i3c controller halts in case of the following conditions:

- Receiving NACK for the Address header of the private transfers if 'IBA_INCLUDE' bit is enabled in the 'DEVICE_CTRL' register.
- Receiving NACK for the Slave address of the private transfers.

The controller updates the 'ERR_STS' field with appropriate error information in the Response status and halts the controller and gives back the control to the application to resume the operation of the controller through writing '1' to the Resume bit of the DEVICE_CTRL register.

2.6.8.1.7 I2C Private Write or Read Transfers

The I2C private write and read transfers are initiated on the bus based on the COMMAND_QUEUE_PORT settings as shown in [Table 2-22](#).

Table 2-22 I2C Private Write/Read Transfers Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	0	Indicates the controller to not consider the CMD field.
	CMD[14]	NA	This field is not applicable since CP bit is set to 0
	CMD[13:7]	NA	This field is not applicable since CP bit is set to 0
	DEV_INDX	DEV_INDX	Indicates the Index of the Device Table which consists of the slave address to be targeted.
	SPEED	0 or 1	Indicates the controller that the transfer should go in I2C SDR mode. Values(I2C mode): <ul style="list-style-type: none"> ■ 0x0: I2C Fm ■ 0x1: I2C Fm+
	SDAP	0 or 1	<ul style="list-style-type: none"> ■ 0: Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0. ■ 1: Indicates the controller to consider the transmit data from the command if RnW is set to 0.
	RnW (Read and Write)	0 or 1	<ul style="list-style-type: none"> ■ 0: Indicates the transfer is write transfer. ■ 1: Indicates the transfer is read transfer.
	PEC	0 or 1	Indicates whether Packet Error Check is enabled for Private SDR transfers. <ul style="list-style-type: none"> ■ 0: PEC check is disabled. ■ 1: PEC check is enabled.
Transfer Argument (SDAP=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Short Data Argument (SDAP=1)	BYTE_STRB	0,1,3,7	Indicates the respective data bytes of the Immediate command are valid.



- The ‘LEGACY_I2C_DEVICE’ bit in the Device Address Table pointed by the ‘DEV_INDX’ field of the Transfer Command must be set to ‘1’ for the I2C private transfers.
- The DWC_mipi_i3c controller uses the I2C protocol to initiate the I2C transfers for the Legacy I2C devices.
- To allow the priority for the IBI from the I3C devices, the DWC_mipi_i3c controller can include Address header for the I2C private transfers through enabling the ‘IBA_INCLUDE’ bit in the ‘DEVICE_CTRL’ register.

The DWC_mipi_i3c controller halts in case of the following conditions:

- Receiving NACK for the Address header of the private transfers if ‘IBA_INCLUDE’ bit is enabled in the ‘DEVICE_CTRL’ register.
- Receiving NACK for the Slave address of the private transfers.

The controller updates the ‘ERR_STS’ field with appropriate error information in the Response status and halts the controller and gives back the control to the application to resume the operation of the controller through writing ‘1’ to the RESUME bit of the DEVICE_CTRL register.

2.6.8.1.8 Implication of Tx-FIFO Empty and Rx-FIFO Full Conditions

At SDR speeds, the DWC_mipi_i3c controller extends the clock by pulling the SCL low under the following conditions:

- Tx-FIFO empty during the middle of Write Data Transfer at SDR Speeds.
- Rx-FIFO full during the middle of Read Data Transfer at SDR Speeds.

2.6.8.1.9 Implication of TOC and ROC Bit Settings for SDR Transfers

The Termination on Completion (TOC) of Transfer Command is used to decide whether to generate a STOP or continue with the next transfer.

- If TOC bit is set to ‘1’, then DWC_mipi_i3c controller generates STOP condition after the end of the transfer.
- If TOC bit is set to ‘0’, then DWC_mipi_i3c controller continues the next pipelined transfer based on the current transfer type:
 - If the current transfer is either Broadcast CCC write or private transfer, then DWC_mipi_i3c controller generates RESTART after the end of the transfer.
 - If the current transfer is the Directed CCC transfer, then DWC_mipi_i3c controller generates RESTART followed by a reserved address (0x7E) after the end of the transfer.
- If TOC bit is set to ‘0’ and the next pipelined command is not available, then DWC_mipi_i3c controller extends the clock by pulling SCL low until the next command is available.

The Response on completion (ROC) bit of the Transfer command is used to generate the Response status of the transfer after the end of the transfer. The Response status is automatically generated if any error condition is occurred for the transfer irrespective of the ROC bit.

2.6.8.1.10 High Data Rate (HDR) Transfers

The High Data Rate transfer protocols are different from SDR protocol and is used to transfer more data at the same SDR frequency. The HDR-Modes have Bus-wide effect and Master is required to enter an HDR mode through CCC before initiating a transfer to the slave. The slaves decode the incoming CCC and enter the HDR mode for the reception of the transfer. The following HDR modes are supported by the DWC_mipi_i3c controller.

- HDR-DDR (Double Data Rate) mode.
- HDR-TSP/L (Ternary symbol Pure or Legacy inclusive bus)

2.6.8.1.11 High Data Rate-Double Data Rate (HDR-DDR) Transfers

The HDR-DDR write and read transfers are initiated on the bus based on the COMMAND_QUEUE_PORT and settings shown in [Table 2-23](#).

Table 2-23 HDR-DDR Transfer Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	1	Indicates the controller to consider the CMD field.
	CMD[13:7]	0x00 – 0xFF	This field indicates either Write or Read command used in the command code of HDR Transfer. <ul style="list-style-type: none"> ■ 0x00 – 0x1F: I3C Reserved Write commands ■ 0x20 – 0x7F: I3C Vendor Write commands ■ 0x80 – 0x9F: I3C Reserved Read Commands ■ 0xA0 – 0xFF: I3C vendor Read Commands
	DEV_INDX	DEV_INDX	Indicates the Index of the Device Table which consists of the slave address to be targeted.
	SPEED	6 (HDR-DDR)	Indicates the controller that the transfer should go in HDR-DDR mode.
	SDAP	0	0 - Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0.
	RnW (Read and Write)	0 or 1	<ul style="list-style-type: none"> ■ 0 - Indicates the transfer is write transfer. ■ 1 - Indicates the transfer is read transfer. Note: This Bit is used for the Write or Read command used in HDR-CMD (CMD[14]) field.
Transfer Argument (SDAP=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.



Note The HDR Transfers are supported only with ‘Transfer Argument command’ and not with ‘Short Data Argument command’ since at the most two words can be transferred in Short Data Argument command format and it really does not justify the overheads in terms of ENTHDR* CCC, HDR Command, entry and exit patterns.

The DWC_mipi_i3c controller generates the ‘ENTHDR0’ CCC to enter into the HDR-DDR mode and forms the Command code through the CMD field of Transfer command and slave address. The Preamble and parity bits are generated by the controller as per the protocol and sends along with the write data word. The DWC_mipi_i3c considers Word from the Tx-FIFO and appends the parity bits and preamble bits to form the Write Data word and transmits the data. The HDR-RESTART and HDR-EXIT pattern is generated in place of RESTART and STOP condition.

To avoid the initial latencies of the transfer, the controller uses ‘TX/RX_START_THLD’ before initiating the transfer. ‘TX_START_THLD’ indicates whether the threshold level of data is present in the Transmit buffer for write transfer or ‘RX_START_THLD’ level of space is available in the Receive buffer for the Read transfer before initiating the transfer. This threshold is only applicable for the transfers which are initiated with the START or RESTART condition. If the Threshold amount of Write data in the Tx-FIFO or empty space in Rx-FIFO is not available, the controller waits until the threshold amount of data is available to initiate the transfer. If the threshold amount of data is not available for the HDR-DDR transfer initiated, which must be initiated with RESTART, then the controller generates EXIT pattern followed by STOP for the current transfer and waits for the threshold amount of data and initiate the next transfer with a START condition.

The DWC_mipi_i3c controller halts in case of the following conditions:

- Receiving NACK for the Address header of the private transfers.
- Receiving NACK for the Slave address (HDR-CMD) of the HDR-DDR private transfers.
- If the controller experiences the Transfer Underflow or Receive Overflow during the HDR-DDR transfers.
- After executing software initiated abort.
- The Controller decodes the Parity bits, CRC byte, Frame mismatch and validates them for the Read transfer and report in the ‘ERR_STS’ field of the Response status if there is any error.

The controller updates the ‘ERR_STS’ field with appropriate error information in the Response status and halts the controller and gives back the control to the application to resume the operation of the controller through writing ‘1’ to the Resume bit of the DEVICE_CTRL register.

2.6.8.1.12 High Data Rate-Ternary Symbol (HDR-TSP / HDR-TSL) Transfers

The HDR-TSP/TSL write and read transfers are initiated on the bus based on the COMMAND_QUEUE_PORT and settings as shown in [Table 2-24](#).

Table 2-24 HDR-TSP/L Transfer Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	1	Indicates the controller to consider the CMD field.
	CMD[13:7]	0x00 - 0xFF	This field indicates either Write or Read command used in the command code of HDR Transfer. <ul style="list-style-type: none"> ■ 0x00 – 0x7F: 128 Write commands ■ 0x80 – 0xFF: 128 Read commands
	DEV_INDX	DEV_INDX	Indicates the Index of the Device Table which consists of the slave address to be targeted.
	SPEED	5 (HDR-TSP/L)	Indicates the controller that the transfer should go in HDR-TSP/L mode.
	SDAP	0	0 - Indicates to consider the transmit data from the Transmit FIFO if Rnw is set to 0.
	RnW (Read and Write)	0 or 1	<ul style="list-style-type: none"> ■ 0 - Indicates the transfer is write transfer. ■ 1 - Indicates the transfer is read transfer. Note: This Bit should match with the Write or Read command used in CMD field.
Transfer Argument (SDAP=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.



The HDR Transfers are supported only with ‘Transfer Argument command’ and not with ‘Short Data Argument command’ since at the most two words can be transferred in Short Data Argument command format and it really does not justify the overheads in terms of ENTHDR* CCC, HDR Command, entry and exit patterns

The DWC_mipi_i3c controller generates the ‘ENTHDR1’ or ‘ENTHDR2’ CCC based on the ‘LEGACY_I2C_DEVICE’ setting in the DEVICE_CTRL register to enter the respective HDR mode and forms the Command code through the CMD field of Transfer command and slave address. Parity bits are generated by the controller and appended with the data word from the application. It is then encoded to Ternary code and transmitted on the bus. The Controller automatically generates the dummy symbols in the ternary code for the HDR-TSL mode.

To avoid the initial latencies of the transfer, the controller uses ‘TX/RX_START_THLD’ before initiating the transfer. ‘TX_START_THLD’ indicates whether the threshold level of data is present in the Transmit buffer for write transfer or ‘RX_START_THLD’ level of space is available in the Receive buffer for the Read

transfer before initiating the transfer. This threshold applies for the transfers, which are initiated with the START or RESTART condition.

If the threshold amount of data is not available for the HDR-DDR transfer that must be initiated with RESTART, then the controller generates EXIT pattern followed by STOP for the current transfer and waits for the threshold amount of data and initiate the next transfer with a START condition.

Per I3C specification, both SCL and SDA must be sampled within the same symbol time. This makes it imperative that the Recovered Clock from the external CDR be available to the controller within the Symbol time itself. Consequence of delays beyond Symbol Time from the CDR is that the Master does not detect Bus Turn around and starts driving within the stipulated time overlapping with the drive from the slave. If the Master is unable to detect three consecutive 'SYM-2' and start driving before 'tEdge' time

There is a period of time where the slave releases SCL and SDA and the Master does not start driving. This leads to a glitch on the bus. The glitch distorts the well defined Exit pattern slaves look out for and can lead to slaves not recognizing HDR Exit and possibly undefined behavior by the slaves.

The Controller has a register field to make a larger time budget available for the external CDR during Bus Turnaround, that is, the low period after 3 Consecutive 'Symbol-2's.

The Master controller detects two Consecutive 'Symbol-2's in Recovered Clock Domain(hdr_clk) and the 3Rd 'Symbol-2' directly from incoming SDA(core clock domain) and starts driving after a fixed three core clocks for synchronization + tskew(max 12.8) from detection of the 3Rd Symbol 2.

The dependency on Recovered Clock is avoided for the last Symbol 2 detection. This allows the Master to Detect Turnaround and start driving before the slave releases the line. Note that the Delay in CDR giving the Recovered Clock is still expected to be < Symbol Time.

SCL_EXT_TERNIN_LCNT_TIMING[I3C_TS_SKEW_CNT] is used to mention the skew in terms of core_clk.

The DWC_mipi_i3c controller accepts the recovered clock as an input for the receive ternary symbols which is used to sample the received ternary symbols and for decoding them. The recovered clock 'hdr_clk' can be either generated through Analog CDR or digital CDR which is outside of the DWC_mipi_i3c controller. The controller generates signal 'hdr_ts_rx_en' to enable external clock recovery circuit during HDR-TS Read transfers.

The DWC_mipi_i3c controller halts in case of the following conditions:

- Receiving NACK for the Address header of the private transfers.
- Receiving NACK for the Slave address (HDR-CMD) of the HDR-TSP/L private transfers.
- If the controller experiences the Transfer Underflow or Receive Overflow during the HDR-DDR transfers.
- After executing the software initiated abort.
- The controller decodes the Parity bits, Frame and validates them for the Read transfer and report in the 'ERR_STS' field of the Response status if there is any error.

The controller updates the 'ERR_STS' field with appropriate error information in the Response status and halts the controller and gives back the control to the application to resume the operation of the controller through writing '1' to the Resume bit of the DEVICE_CTRL register.

2.6.8.1.13 Implication of Tx-FIFO Empty and Rx-FIFO Full Conditions

The DWC_mipi_i3c controller terminates the transfer abruptly during the following conditions and reports it in the 'ERR_STS' field of the Response status since the clock extension is not possible in HDR modes.

- Tx-FIFO empty during the middle of Write Data Transfer.
- In HDR-DDR Mode, Rx-FIFO full during the middle of Read Data Transfer through 'Master Abort' feature.
- In HDR-TSP/L Mode, The controller stops receiving the data from the I3C line, if Rx-FIFO is full and ends the transfer only after slave ends the transfer since there is no 'Master Abort' feature in TSP/L Mode.

2.6.8.1.14 Implication of TOC and ROC Bit Settings for HDR Transfers

The Termination on Completion (TOC) of Transfer Command is used to decide whether to generate HDR-EXIT or HDR-RESTART to continue with the next transfer.

- If TOC bit is set to '1', then DWC_mipi_i3c controller generates HDR EXIT after the end of the transfer.
- If TOC bit is set to '0', then DWC_mipi_i3c controller continues the next pipelined transfer based on the current transfer type.

Table 2-25 Implication of TOC and ROC Bit Settings for HDR Modes

Current Transfer	Next Transfer	Start Threshold of Next Transfer	DWC_mipi_i3c Controller Action
HDR-DDR	HDR-DDR	Met	<ol style="list-style-type: none"> 1. Generates HDR-RESTART at the end of transfer. 2. Continue with the HDR command of next transfer.
HDR-DDR	HDR-DDR	Not met	<ol style="list-style-type: none"> 3. Generates HDR-EXIT at the end of transfer. 4. Wait for the Start threshold for the pipelined transfer to be met. 5. Generate the pipelined transfer with START after the start threshold is met.
HDR-DDR	HDR-TSP/L	NA	<ol style="list-style-type: none"> 6. Generates HDR-EXIT at the end of transfer. 7. Wait for Bus-free time. 8. Initiate the next HDR-TSP/L from START condition.
HDR-TSP/L	HDR-TSP/L	Met	<ol style="list-style-type: none"> 9. Generates HDR-RESTART at the end of transfer. 10..Continue with the HDR command of next transfer.

Table 2-25 Implication of TOC and ROC Bit Settings for HDR Modes (Continued)

Current Transfer	Next Transfer	Start Threshold of Next Transfer	DWC_mipi_i3c Controller Action
HDR-TSP/L	HDR-TSP/L	Not met	11. Generates HDR-EXIT at the end of transfer. 12..Wait for the Start threshold for the pipelined transfer to be met. 13..Generate the pipelined transfer with START after the start threshold is met.
HDR-DDR / HDR-TSP/L	SDR	NA	14.. Generates HDR-EXIT at the end of transfer. 15.Wait for Bus-free time. 16.. Initiate the next SDR transfer with START condition.

The Response on Completion (ROC) bit of the Transfer command is used to generate the response status of the transfer after the end of the transfer. The response status is automatically generated if any error condition is occurred for the transfer irrespective of the ROC bit.

2.6.8.2 Bus Communication Protocols in HCI Mode

The DWC_mipi_i3c supports the following bus communication protocols to transfer the data between master and slave.

- Single Data Rate (SDR) Transfers
- High Data Rate -Double Data Rate (HDR-DDR)
- High Data Rate – Ternary Mode (HDR-TSP and HDR-TSL)

2.6.8.2.1 Single Data Rate (SDR) Transfers in Master Mode

In SDR mode, Single bit is sent on each SCL clock. All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer. After the master sends the address and R/W bit or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal (ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high so that the master can abort the transfer.

The single data rate transfers are initiated by the application through the command as explained in “[Single Data Rate \(SDR\) Transfers in Master Mode](#)” on page 91. Based on the command initiated by the application and Device address pointed by the DEV_INDX field of the command, or in case of no DAT or DCT configuration on the SLV_ADDR field present on the queued command, the DWC_mipi_i3c controller initiate the transfers on the bus. [Table 2-26](#) illustrates the decoded transfer type based on Transfer command and Device Address Table or I2CNI3C command field - in case of no DAT or DCT configuration.

Table 2-26 Transfer Type Decoding Based on Command and Device Address Table

Transfer Command (Command Port)				Device Address Table/ I2CnI3C Command Field	
CP	CMD[14]	MODE	RnW	DEVICE Type	Decoded Transfer Type
0	NA	0 to 4	0	0	I3C Private Write Transfer
0	NA	0 to 4	1	0	I3C Private Read Transfer
1	0	0 to 4	0	0	I3C Broadcast CCC Write Transfer
1	1	0 to 4	0	0	I3C Directed CCC Write Transfer
1	1	0 to 4	1	0	I3C Directed CCC Read Transfer
0	x	0 to 4	0	1	I2C Private Write Transfer
0	x	0 to 4	1	1	I2C Private Read Transfer
Others				Illegal Combination	

2.6.8.2.2 Broadcast or Directed CCC Transfers

The Common Command Code transfers are used for the bus management procedures such as the following:

- Receiving the device characteristics
- Setting the max write or read data length
- Enabling or disabling the event generation in the I3C devices
- Informing the bus characteristics to all the existing devices.
- Entering the high data rate transfer modes

2.6.8.2.3 Broadcast CCC Write Transfers

The Broadcast CCC Write transfers are initiated on the bus based on the COMMAND_PORT settings as shown in [Table 2-27](#).

Table 2-27 Broadcast CCC Write Transfer Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	1	Indicates the controller to consider the CMD field.
	CMD[14]	0	Indicates the controller that the transfer is Broadcast CCC transfer.
	CMD[13:7]	0x0 – 0x7F	Indicates the CCC Command to be transferred.
	DEV_INDX/ SLV_ADDR	NA	This field is not used since the transfer is Broadcast CCC.
	MODE	0 (SDR0)	Indicates the controller that the transfer should go in SDR mode. Note: All CCC transfers is initiated with SDR0 Speed.
	CMD_ATTR	0 or 1	<ul style="list-style-type: none"> ■ 0 - Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0. ■ 1 - Indicates the controller to consider the transmit data from the command if RnW is set to 0.
	RnW	0	Indicates the transfer is either write or read transfer. <ul style="list-style-type: none"> ■ 0 – Write Transfer ■ 1 – Read Transfer
Regular Transfer (CMD_ATTR=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Immediate Transfer (CMD_ATTR=1)	BYTE_CNT	0 to 4	Indicates the respective data bytes of Immediate command are valid.

Data in Tx-FIFO (Regular transfer) or data bytes (Immediate Data Transfer) are not required for some broadcast CCC's which does not have payload data indicated through data length (Regular transfer) or BYTE_CNT (Immediate transfer). If the Broadcast CCC does not consist of payload, you must indicate it with zero in either data length or BYTE_CNT fields based on the command issued.

The DWC_mipi_i3c controller halts in case of receiving NACK (It means no I3C device on the bus) for the address header of the Broadcast CCC transfer. The controller updates the 'ERR_STS' field with appropriate error information in the Response status, halts the controller and gives back the control to the application to resume the operation of controller through writing '1' to the RESUME bit of the HC_CONTROL register.

2.6.8.2.4 Directed Write and Read Transfers

The Directed CCC Write/Read transfers are initiated on the bus based on the COMMAND_PORT settings as shown in [Table 2-28](#).

Table 2-28 Directed CCC Write/Read Transfers Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	1	Indicates the controller to consider the CMD field.
	CMD[14]	1	Indicates the controller that the transfer is Directed CCC transfer.
	CMD[13:7]	0x0 - 0x7F	Indicates the CCC Command to be transferred.
	I2CNI3C	0	Indicates if the slave is I2C (1'b1) or I3C (1'b0) only used in configurations with no DAT or DCT reserved otherwise.
	DEV_INDX/ SLV_ADDR	DEV_INDX/ SLV_ADDR	Indicates the Index of the Device Table (IC_HAS_DAT=1) which consists of the slave address to be targeted or the slave address in configurations without DAT or DCT (IC_HAS_DAT=0).
	MODE	0 (SDR0)	Indicates the controller that the transfer should go in SDR mode. Note: All CCC transfers are initiated with SDR0 Speed.
	CMD_ATTR	0 or 1	<ul style="list-style-type: none"> ■ 0 - Indicates to consider the transmit data from the Transmit FIFO if Rnw is set to 0. ■ 1 - Indicates the controller to consider the transmit data from the command if RnW is set to 0.
	RnW	0 or 1	<ul style="list-style-type: none"> ■ 0 - Indicates the transfer is write transfer. ■ 1 – Indicates the transfer is read transfer.
Regular Transfer (CMD_ATTR=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Immediate Transfer (CMD_ATTR=1)	BYTE_CNT	0 to 4	Indicates the respective data bytes of the Immediate command are valid.

The Data in Tx-FIFO (Regular transfer) or data bytes (Immediate data transfer) are not required for some directed write CCC's which do not have payload data indicated through data length (Regular transfer) or BYTE_CNT (Immediate transfer). If the directed CCC does not consist of payload, you must indicate it with zero in either data length or BYTE_CNT fields based on the command issued.

2.6.8.2.5 Directed CCC Transfer Targeted to Multiple Slaves

Each transfer command initiates Directed CCC transfer to only one slave since it consists of only one DEV_INDX/SLV_ADDR field. To transfer the directed CCC command to multiple devices, pipeline the

multiple transfer commands in the COMMAND_PORT with TOC bits set to zero and with the different DEV_INDX or SLV_ADDR fields pointing to multiple Slaves. The DWC_mipi_i3c controller decodes the pipelined ‘Transfer command’ during the transfer of Directed CCC transfer and decides the next transfer based on the following:

- If the current command and the pipelined command has same Directed CCC, then the controller targets the next slave without ending the CCC command.
- If the current command and the pipelined command are not the same Directed CCC, then the controller ends the CCC command and starts issuing the next transfer as indicated by the pipelined transfer command.

The application can set the ROC bit to ‘0’ for the subsequent Directed CCC commands and enable the ROC bit in the last CCC command if Directed CCC transfer is targeted to multiple devices to avoid unnecessary responses.

The DWC_mipi_i3c controller halts in case of the following conditions:

- Receiving NACK for the address header of the Directed CCC transfer (It means no I3C device on the bus)
- Receiving NACK for the Slave address of the Directed CCC transfer

The controller updates the ‘ERR_STS’ field with appropriate error information in the Response status and halt the controller and gives back the control to the application to resume the operation of controller through writing ‘1’ to the Resume bit of the ‘HC_CONTROL’ register.

2.6.8.2.6 I3C Private Write or Read Transfers

The I3C private write and read transfers are initiated on the bus based on the COMMAND_PORT and settings shown in [Table 2-29](#).

Table 2-29 I3C Private Write/Read Transfers Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	0	Indicates the controller to not consider the CMD field.
	CMD[14]	NA	This field is not applicable since CP bit is set to 0
	CMD[13:7]	NA	This field is not applicable since CP bit is set to 0
	I2CNI3C	0	Indicates if the Slave is I2C (1'b1) or I3C (1'b0) only used in configurations with no DAT or DCT reserved otherwise.
	DEV_INDX/ SLV_ADDR	DEV_INDX/ SLV_ADDR	Indicates the Index of the Device Table (IC_HAS_DAT=1) which consists of the slave address to be targeted or the slave address in configurations without DAT or DCT (IC_HAS_DAT=0).
	MODE	0 to 4	Indicates the controller that the transfer should go in SDR mode.
	CMD_ATTR	0 or 1	<ul style="list-style-type: none"> ■ 0 - Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0. ■ 1 - Indicates the controller to consider the transmit data from the command if RnW is set to 0.
	RnW (Read and Write)	0 or 1	<ul style="list-style-type: none"> ■ 0 - Indicates the transfer is write transfer. ■ 1 - Indicates the transfer is read transfer.
Regular Transfer (CMD_ATTR=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Immediate Transfer (CMD_ATTR=1)	BYTE_CNT	0 to 4	Indicates the respective data bytes of the Immediate command are valid.
Short Data Argument (CMD_ATTR=1)	BYTE_STRB	0,1,3,7	Indicates the respective data bytes of the Immediate command are valid.



- For configuration with DAT, the 'DEVICE' bit in the Device Address Table pointed by the 'DEV_INDX' field of the Transfer Command should be set to zero for the I3C private transfers.
- To avoid the initial latencies of the transfer, the controller uses 'TX/RX_START_THLD' before initiating the transfer. 'TX_START_THLD' ensures that the threshold level of data is present in the Transmit buffer for write transfer and 'RX_START_THLD' level of space is available in the Receive buffer for the Read transfer before initiating the transfer. This threshold is applicable only for the transfers which are initiated with the START condition and not applicable for the transfers which are initiated with the RESTART condition for SDR transfers.
- To allow the priority for IBI from the devices, the DWC_mipi_i3c controller can include address header for the I3C private transfers by enabling the 'IBA_INCLUDE' bit in the 'HC_CONTROL' register.

The DWC_mipi_i3c controller halts in case of the following conditions:

- Receiving NACK for the Address header of the private transfers if 'IBA_INCLUDE' bit is enabled in the 'HC_CONTROL' register.
- Receiving NACK for the Slave address of the private transfers.

The controller updates the 'ERR_STS' field with appropriate error information in the Response status and halts the controller and gives back the control to the application to resume the operation of the controller by writing '1' to the Resume bit of the HC_CONTROL register.

2.6.8.2.7 I2C Private Write or Read Transfers

The I2C private write and read transfers are initiated on the bus based on the COMMAND_PORT settings as shown in [Table 2-30](#).

Table 2-30 I2C Private Write/Read Transfers Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	0	Indicates the controller to not consider the CMD field.
	CMD[14]	NA	This field is not applicable since CP bit is set to 0
	CMD[13:7]	NA	This field is not applicable since CP bit is set to 0
	I2CNI3C	1	Indicates if the Slave is I2C (1'b1) or I3C (1'b0) only used in configurations with no DAT or DCT reserved otherwise.
	DEV_INDX/ SLV_ADDR	DEV_INDX/ SLV_ADDR	Indicates the Index of the Device Table (IC_HAS_DAT=1) which consists of the slave address to be targeted or the slave address in configurations without DAT or DCT (IC_HAS_DAT=0).
	MODE	0 to 2	Indicates the controller that the transfer should go in I2C SDR mode.
	CMD_ATTR	0 or 1	<ul style="list-style-type: none"> ■ 0 - Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0. ■ 1 - Indicates the controller to consider the transmit data from the command if RnW is set to 0.
	RnW (Read and Write)	0 or 1	<ul style="list-style-type: none"> ■ 0 - Indicates the transfer is write transfer. ■ 1 - Indicates the transfer is read transfer.
Regular Transfer (CMD_ATTR=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Immediate Transfer (CMD_ATTR=1)	BYTE_CNT	0 to 4	Indicates the respective data bytes of the Immediate command are valid.


Note

- For configurations with DAT, the 'DEVICE' bit in the Device Address Table pointed by the 'DEV_INDX' field of the Transfer Command should be set to '1' for the I2C private transfers.
- The DWC_mipi_i3c controller uses the I2C protocol to initiate I2C transfers for the legacy I2C devices.
- To allow the priority for the IBI from the I3C devices, the DWC_mipi_i3c controller can include address header for the I2C private transfers by enabling the 'IBA_INCLUDE' bit in the 'HC_CONTROL' register.

The DWC_mipi_i3c controller halts in the following conditions:

- Receiving NACK for the Address header of the private transfers if 'IBA_INCLUDE' bit is enabled in the 'HC_CONTROL' register.
- Receiving NACK for the Slave address of the private transfers.

The controller updates the ‘ERR_STS’ field with appropriate error information in the Response status and halts the controller and gives back the control to the application to resume the operation of the controller by writing ‘1’ to the RESUME bit of the DEVICE_CTRL register.

2.6.8.2.8 Implication of Tx-FIFO Empty and Rx-FIFO Full Conditions

At SDR Speeds, the DWC_mipi_i3c controller extends the clock by pulling the SCL low under the following conditions:

- Tx-FIFO empty during the middle of a write data transfer at SDR speeds.
- Rx-FIFO full during the middle of a read data transfer at SDR speeds.

2.6.8.2.9 Implication of TOC and ROC Bit Settings for SDR Transfers

The Termination on Completion (TOC) of transfer command is used to decide whether to generate STOP or continue with the next transfer.

- If TOC bit is set to ‘1’, then DWC_mipi_i3c controller generates STOP condition after the end of the transfer.
- If TOC bit is set to ‘0’, then DWC_mipi_i3c controller continues the next pipelined transfer based on the current transfer type:
 - If the current transfer is either Broadcast CCC write or private transfer, then DWC_mipi_i3c controller generates RESTART after the end of the transfer.
 - If the current transfer is Directed CCC transfer, then DWC_mipi_i3c controller generates RESTART followed by Reserved address(0x7E) after the end of the transfer.
- If TOC bit is set to ‘0’ and the next pipelined command is not available, then DWC_mipi_i3c controller extends the clock through SCL pulling low until the next command is available.

The Response on completion (ROC) bit of the Transfer command is used to generate the Response status of the transfer after the end of the transfer. The Response status is automatically generated if any error condition occurs for the transfer irrespective of the ROC bit.

2.6.8.2.10 High Data Rate (HDR) Transfers

The High Data Rate transfer protocol is different from SDR protocol and is used to transfer more data at the same SDR frequency. The HDR-modes have bus-wide effect and Master is required to enter HDR mode through CCC before initiating transfer to the Slave. The Slaves decode the incoming CCC and enter in to the respective HDR mode for the reception of the transfer. The following HDR modes are supported by the DWC_mipi_i3c controller.

- HDR-DDR (Double Data Rate) mode
- HDR-TSP/L (Ternary symbol Pure or Legacy inclusive bus)

2.6.8.2.11 High Data Rate-Double Data Rate (HDR-DDR) Transfers

The HDR-DDR write and read transfers are initiated on the bus based on the COMMAND_PORT and the settings shown in [Table 2-31](#).

Table 2-31 HDR-DDR Transfer Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	1	Indicates the controller to consider the CMD field.
	CMD[13:7]	0x00 – 0xFF	This field indicates either Write or Read command used in the command code of HDR Transfer. 0x00 – 0x1F: I3C Reserved Write commands 0x20 – 0x7F: I3C Vendor Write commands 0x80 – 0x9F: I3C Reserved Read Commands 0xA0 – 0xFF: I3C vendor Read Commands
	I2CNI3C	0	Indicates if the Slave is I2C (1'b1) or I3C (1'b0) only used in configurations with no DAT or DCT reserved otherwise.
	DEV_INDX/ SLV_ADDR	DEV_INDX/ SLV_ADDR	Indicates the Index of the Device Table (IC_HAS_DAT=1), which consists of the slave address to be targeted or the slave address in configurations without DAT or DCT (IC_HAS_DAT=0).
	MODE	6 (HDR-DDR)	Indicates the controller that the transfer should go in HDR-DDR mode.
	CMD_ATTR	0	0 - Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0.
	RnW (Read and Write)	0 or 1	0 - Indicates the transfer is write transfer. 1 - Indicates the transfer is read transfer. Note: This Bit is used for the Write or Read command used in HDR-CMD (CMD[14]) field.
Regular Transfer (CMD_ATTR=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.



The HDR Transfers are supported only with ‘Regular Transfer command’ and not with ‘Immediate Transfer command’ since at the most two words can be transferred in Immediate Transfer command format.

The DWC_mipi_i3c controller generates ‘ENTHDR0’ CCC to enter into the HDR-DDR mode and forms the command code through CMD field of transfer command and Slave address. The Preamble and parity bits are generated by the controller as per the protocol and sends along with the write data word. The DWC_mipi_i3c controller considers Word from the Tx-FIFO and appends the parity bits and preamble bits to form the write data word and transmits the data. The HDR-RESTART and HDR-EXIT pattern is generated in place of RESTART and STOP condition.

To avoid the initial latencies of the transfer, the controller uses ‘TX/RX_START_THLD’ before initiating the transfer. ‘TX_START_THLD’ indicates whether the threshold level of data is present in the transmit buffer

for write transfer or ‘RX_START_THLD’ level of space is available in the receive buffer for the read transfer before initiating the transfer. This threshold is only applicable for the transfers which are initiated with the START or RESTART condition. If the threshold amount of write data in Tx-FIFO or empty space in Rx-FIFO is not available, the controller waits until the threshold amount of data is available to initiate the transfer. If the threshold amount of data is not available for the HDR-DDR transfer initiated which has to be initiated with RESTART, then the controller generates EXIT pattern and waits for the threshold amount of data to initiate the transfer.

The DWC_mipi_i3c controller halts in case of the following conditions:

- Receiving NACK for the address header of the private transfers if ‘IBA_INCLUDE’ bit is enabled in the ‘HC_CONTROL’ register.
- Receiving NACK for the Slave address (HDR-CMD) of the HDR-DDR private transfers.
- If the controller experiences the transfer underflow or receive overflow during the HDR-DDR transfers.
- After executing software initiated abort.
- The controller decodes the parity bits, CRC byte, Frame mismatch and validates them for the read transfer and reports in the ‘ERR_STS’ field of the response status if there is any error.

The controller updates the ‘ERR_STS’ field with appropriate error information in the response status and halts the controller and gives back the control to the application to resume the operation of controller through writing ‘1’ to the Resume bit of the HC_CONTROL register.

2.6.8.2.12 High Data Rate-Ternary Symbol (HDR-TSP / HDR-TSL) Transfers

The HDR-TSP/TSL write and read transfers are initiated on the bus based on the COMMAND_PORT and settings as shown in [Table 2-32](#).

Table 2-32 HDR-TSP/L Transfer Required Programming Values

Command Attribute	Bit-Field	Programmed Value	Description
Transfer Command	CP	1	Indicates the controller to consider the CMD field.
	CMD[13:7]	0x00 - 0xFF	This field indicates either Write or Read command used in the command code of HDR Transfer. 0x00 – 0x7F: I28 Write commands 0x80 – 0xFF: 128 Read commands
	I2CNI3C	0	Indicates if the slave is I2C (1'b1) or I3C (1'b0) only used in configurations with no DAT or DCT, reserved otherwise.
	DEV_INDX/SLV_ADDR	DEV_INDX/SLV_ADDR	Indicates the Index of the Device Table (IC_HAS_DAT=1) which consists of the slave address to be targeted or the slave address in configurations without DAT or DCT (IC_HAS_DAT=0).
	MODE	5 (HDR-TSP/L)	Indicates the controller that the transfer should go in HDR-TSP/L mode.
	CMD_ATTR	0	0 - Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0.
	RnW (Read and Write)	0 or 1	<ul style="list-style-type: none"> ■ 0 - Indicates the transfer is write transfer. ■ 1 - Indicates the transfer is read transfer. <p>Note: This Bit should match with the Write or Read command used in CMD field.</p>
Regular Transfer (CMD_ATTR=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.



The HDR transfers are supported only with 'Regular Transfer command' and not with 'Immediate Transfer command' since at the most two words can be transferred in Immediate transfer command format.

The DWC_mipi_i3c controller generates 'ENTHDR1' or 'ENTHDR2' CCC based on the 'I2C_SLAVE_PRESENT' setting in the HC_CONTROL register to enter the respective HDR mode and forms the command code through CMD field of transfer command and Slave address. The parity bits are generated by the controller and appends with the data word from the application and encodes to Ternary code and transmits on the bus. The controller automatically generates the dummy symbols in the ternary code for the HDR-TSL mode.

To avoid the initial latencies of the transfer, the controller uses 'TX/RX_START_THLD' before initiating the transfer. 'TX_START_THLD' indicates whether the threshold level of data is present in the transmit buffer for write transfer or 'RX_START_THLD' level of space is available in the receive buffer for the read transfer before initiating the transfer. This threshold is applicable only for the transfers which are initiating with the

START or RESTART condition. If the threshold amount of write data in Tx-FIFO or empty space in Rx-FIFO is not available, the controller waits until the threshold amount of data is available to initiate the transfer. If the threshold amount of data is not available for the HDR-TSP/L transfer initiated which has to be initiated with RESTART, then the controller generates EXIT pattern and waits for the threshold amount of data to initiate the transfer.

The DWC_mipi_i3c controller expects the recovered clock as an input for the receive ternary symbols which is used to sample the received ternary symbols and for decoding them. The recovered clock 'hdr_clk' can be either generated through analog CDR or digital CDR which is outside of the DWC_mipi_i3c controller.

The DWC_mipi_i3c controller halts in case of the following conditions:

- Receiving NACK for the address header of the private transfers if 'IBA_INCLUDE' bit is enabled in the 'HC_CONTROL' register.
- Receiving NACK for the Slave address (HDR-CMD) of the HDR-TSP/L private transfers.
- If the controller experiences the transfer underflow or receive overflow during the HDR-DDR transfers.
- After executing Software Initiated Abort.
- The controller decodes the parity bits, Frame and validates them for the read transfer and reports in the 'ERR_STS' field of the Response status if there is any error.

The controller updates the 'ERR_STS' field with appropriate error information in the Response status and halts the controller and gives back the control to the application to resume the operation of controller through writing '1' to the Resume bit of the HC_CONTROL register.

2.6.8.2.13 Implication of Tx-FIFO Empty and Rx-FIFO Full Conditions

The DWC_mipi_i3c controller terminates the transfer abruptly during the following conditions and reports it in the 'ERR_STS' field of the Response status since the clock extension is not possible in HDR modes:

- Tx-FIFO empty during the middle of write data transfer.
- In HDR-DDR mode, Rx-FIFO full during the middle of read data transfer through 'Master Abort' feature.
- In HDR-TSP/L Mode, the controller stops receiving the data from the I3C line, if Rx-FIFO is full and ends the transfer only after the Slave ends the transfer since there is no 'Master Abort' feature in TSP/L Mode.

2.6.8.2.14 Implication of TOC and ROC Bit Settings for HDR Transfers

The Termination on Completion (TOC) of transfer command is used to decide whether to generate HDR-EXIT or HDR-RESTART to continue with the next transfer.

- If TOC bit is set to '1', then DWC_mipi_i3c controller generates HDR EXIT after the end of the transfer.
- If TOC bit is set to '0', then DWC_mipi_i3c controller continues the next pipelined transfer based on the current transfer type.

Table 2-33 Implication of TOC and ROC Bit Settings for HDR Modes

Current Transfer	Next Transfer	Start Threshold of Next Transfer	DWC_mipi_i3c Controller Action
HDR-DDR	HDR-DDR	Met	<ul style="list-style-type: none"> ▪ Generates HDR-RESTART at the end of transfer. ▪ Continue with the HDR command of next transfer.
HDR-DDR	HDR-DDR	Not met	<ul style="list-style-type: none"> ▪ Generates HDR-EXIT at the end of transfer. ▪ Wait for the Start threshold for the pipelined transfer to be met. ▪ Generate the pipelined transfer with START after the start threshold is met.
HDR-DDR	HDR-TSP/L	NA	<ul style="list-style-type: none"> ▪ Generates HDR-EXIT at the end of transfer. ▪ Wait for Bus-free time. ▪ Initiate the next HDR-TSP/L from START condition.
HDR-TSP/L	HDR-TSP/L	Met	<ul style="list-style-type: none"> ▪ Generates HDR-RESTART at the end of transfer. ▪ Continue with the HDR command of next transfer.
HDR-TSP/L	HDR-TSP/L	Not met	<ul style="list-style-type: none"> ▪ Generates HDR-EXIT at the end of transfer. ▪ Wait for the Start threshold for the pipelined transfer to be met. ▪ Generate the pipelined transfer with START after the start threshold is met.
HDR-DDR / HDR-TSP/L	SDR	NA	<ul style="list-style-type: none"> ▪ Generates HDR-EXIT at the end of transfer. ▪ Wait for Bus-free time. ▪ Initiate the next SDR transfer with START condition.

The Response on completion (ROC) bit of the transfer command is used to generate the response status of the transfer after the end of the transfer. The response status is automatically generated if any error condition occurs for the transfer, irrespective of the ROC bit.

2.6.8.3 Clock Stalling Conditions

The Master controller stalls SCL clock during specific scenarios of non-HDR transfers. This is to accommodate intermittent system latencies during command pipelining, transmit data pre-fetching, response reading, and so on. The DWC_mipi_i3c controller also provides additional knobs in the form of Start Thresholds and Buffer Thresholds to avoid clock stalling.

It is recommended that the system design should allocate sufficient bandwidth to the controller's application to provide the transmit data and consume the received data to sustain the maximum supported data rate. At any point in time the system should not exceed the maximum specified latency. Furthermore, stalling is not supported in HDR modes. By design, the system has to meet this requirement in case of HDR transfers failing which it experiences recurrent underruns and overflows.

Clock stalling is a feature that helps avoid overrun and underrun errors in case of SDR notwithstanding all of the mentioned measures. The host controller software receiving these errors is an indication of a larger

issue in itself and neither clock stalling by itself or termination of transfers during clock stalling is the solution.

The FIFO size should be twice of what is likely to be the maximum value of threshold. This facilitates smooth and seamless handshake between software and hardware.

For example, if the maximum value of threshold is determined to be four locations (that is, the level at which the software gets an interrupt be set to 4), it is recommended that the RX buffer depth to be 8 or more. With this, the software is interrupted when the controller has four locations of data to be read out and it still has four more locations free to be able to receive new data if it arrives while the software is reading out data that was received earlier without the risk of the buffers becoming full.

FIFOS (Rx, TX, CMD, IBI, RESP) becoming full while transfers are in progress is a an undesirable situation and it pushes the controller to enter clock stalling, which itself is discouraged by the specification.

Table 2-34 captures various scenarios in which the controller stalls the clock during the SCL Low period.

Table 2-34 Master Clock Stall Conditions

SI No.	Transfer Command	Previous Command Condition	Condition to Enter Clock Stalling
I3C/I2C Transfer, ACK/NACK Phase			
1	Write Transfer Regular command	TOC bit set to '0'	Tx-FIFO is empty
2	Read Transfer Regular command	TOC bit set to '0'	Rx-FIFO is Full
3	Follow-up Directed CCC command without payload (Immediate/Regular) (Not 1st Directed CCC command)	Previous Directed CCC command TOC bit set to '0'	CMD-QUEUE is empty.
4	Broadcast CCC Transfer with Regular command	TOC bit set to '0'	TX-FIFO is empty
5	Directed CCC Write Transfer Regular command	TOC bit set to '0'	TX-FIFO is empty
6	Directed CCC Read Transfer Regular command	TOC bit set to '0'	Rx-FIFO is Full.
7	Directed CCC command without Payload and ROC Bit is set to '1'	TOC bit set to '0'	RESP-Queue is Full
8	Middle of I2C Write Transfer with Regular Transfer command	NA	TX-FIFO is empty
9	Middle of I2C Write Transfer with Regular Transfer command	NA	Rx-FIFO is Full
10	End of I2C Write Transfer Regular command (Only Master Terminates) and TOC bit set to '0'.	NA	Next command unavailable
11	End of I2C Write Transfer Regular command (Only Master Terminates) and ROC bit is set to '1'.	NA	RESP-Queue is Full
12	End of I2C Read Transfer Regular command (Either Master/slave terminates) and TOC bit set to '0'.	NA	Next command unavailable

Table 2-34 Master Clock Stall Conditions (Continued)

SI No.	Transfer Command	Previous Command Condition	Condition to Enter Clock Stalling
13	End of I2C Read Transfer Regular command (Only Master Terminates) and ROC bit is set to '1'.	NA	RESP-Queue is Full
Write Data Transfer, Parity Bit			
14	Middle of Write Transfer Regular command	NA	TX-FIFO is empty
15	End of Write Transfer Regular command (Only Master Terminates) and TOC bit set to '0'.	NA	Next command unavailable
16	End of Write Transfer Regular command (Only Master Terminates) and ROC bit is set to '1'.	NA	RESP-Queue is Full
I3C Read Transfer, Transition Bit			
17	Middle of Read Transfer Regular command	NA	RX-FIFO is Full
18	End of Read Transfer Regular command (Either Master/slave terminates) and TOC bit set to '0'.	NA	Next command unavailable
19	End of Read Transfer Regular command (Only Master Terminates) and ROC bit is set to '1'.	NA	RESP-Queue is Full
I3C IBI Transfer, Transition Bit			
20	Middle of IBI Read Data Transfer	NA	IBI-Data FIFO is Full
21	Middle of Auto Command Read Data Transfer	NA	IBI-Data FIFO is Full

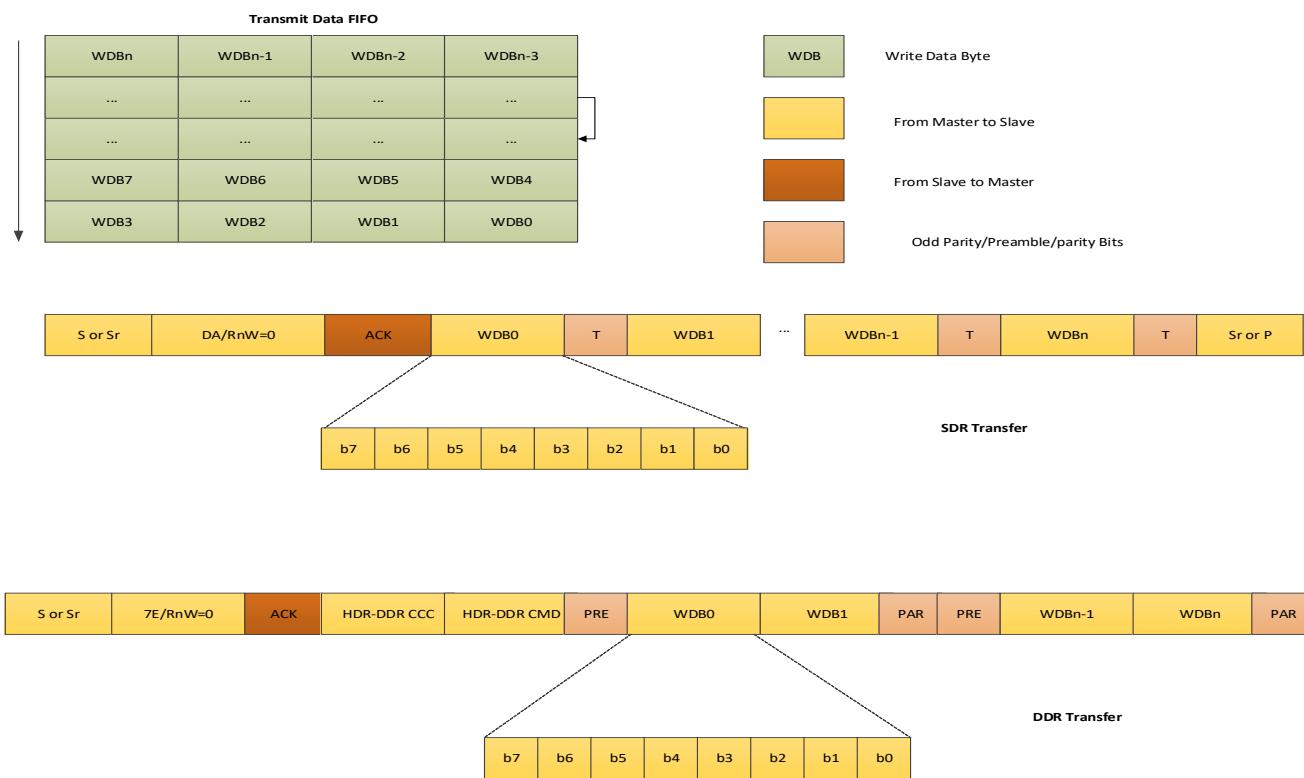
The DWC_mipi_i3c controller exits the clock stalling state once the condition mentioned in ‘condition to enter clock stalling’ column becomes invalid.

2.6.8.4 Data Packing and Unpacking

This section describes the Data packing (Rx-FIFO) and Data unpacking (Tx-FIFO) during the transfers.

2.6.8.4.1 Transmission of Data by DWC_mipi_i3c

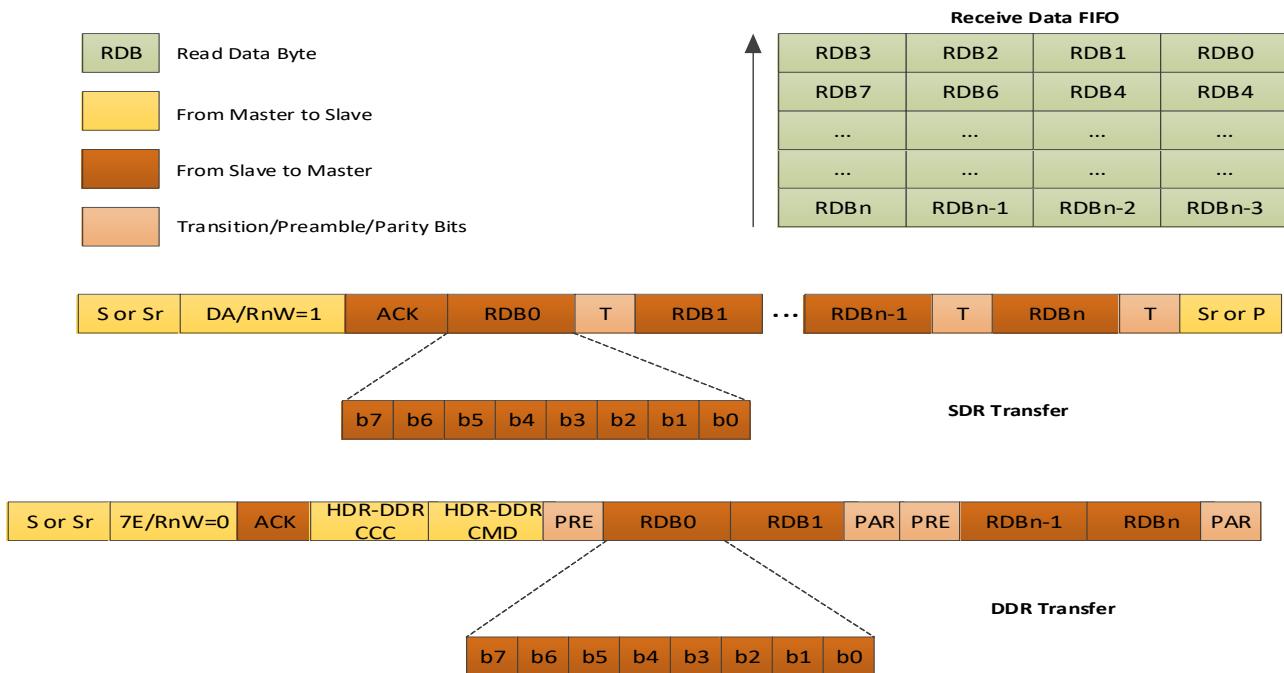
The data to be transmitted is expected to be provided to the controller in the Transmit FIFO, as shown in [Figure 2-23](#)

Figure 2-23 Byte Ordering -Transmit Data

The controller puts the Least Significant Byte of data as the first byte on the I3C bus, followed by successive bytes.

2.6.8.4.2 Packing of Private Read Data by DWC_mipi_i3c into Receive FIFO

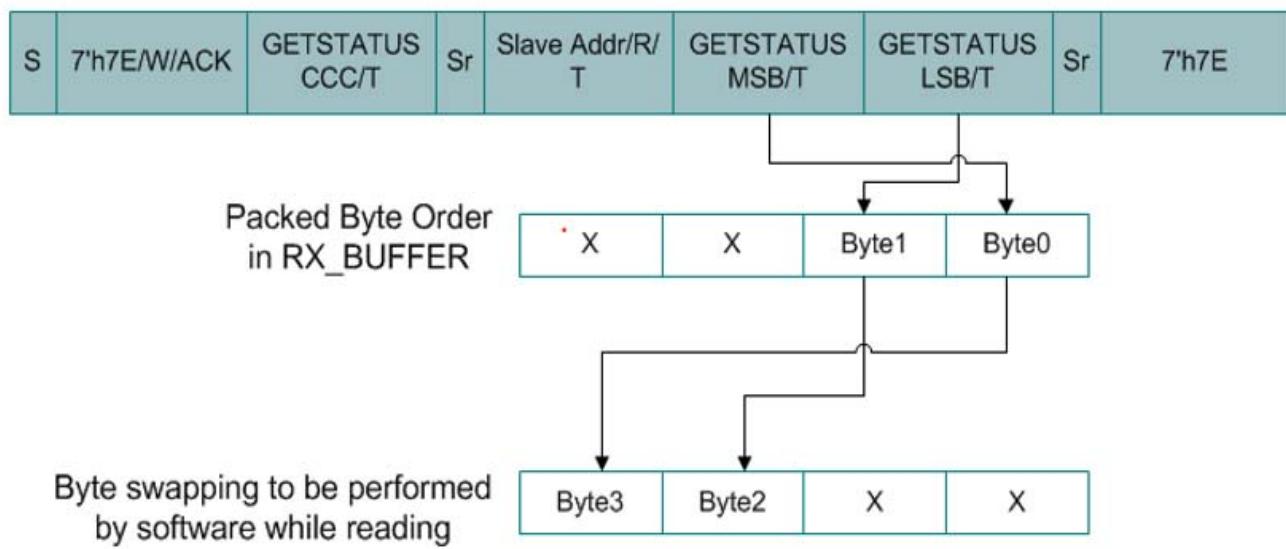
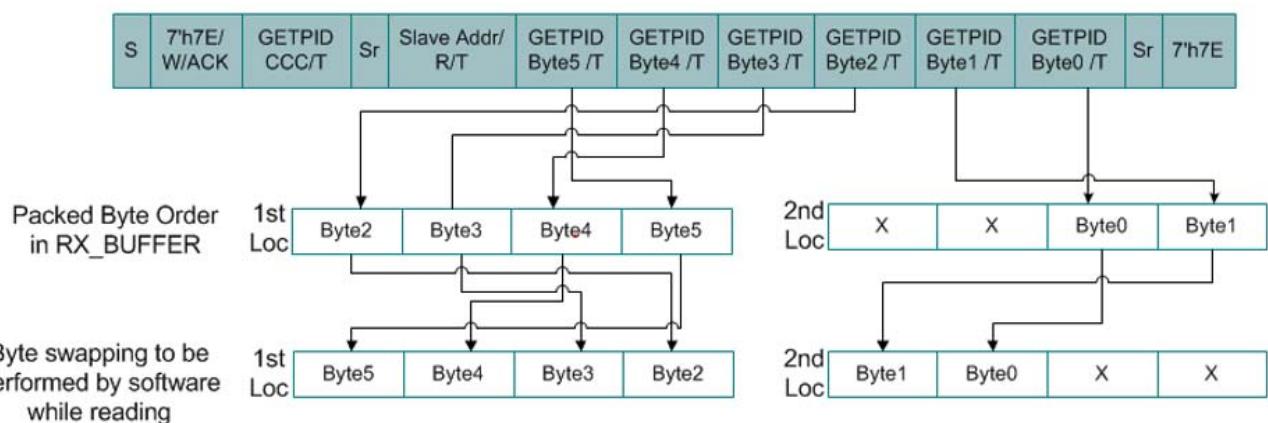
The controller receives the bytes in the same order as on the bus and places the first byte received in the Least Significant Byte (LSB) position, and the following bytes in the LSB + 1 location, until all the four bytes of a word are received in the Receive FIFO. This is shown in [Figure 2-24](#).

Figure 2-24 Byte Ordering - Receive Data

2.6.8.4.3 Packing of Any Read Data by DWC_mipi_i3c into Receive FIFO

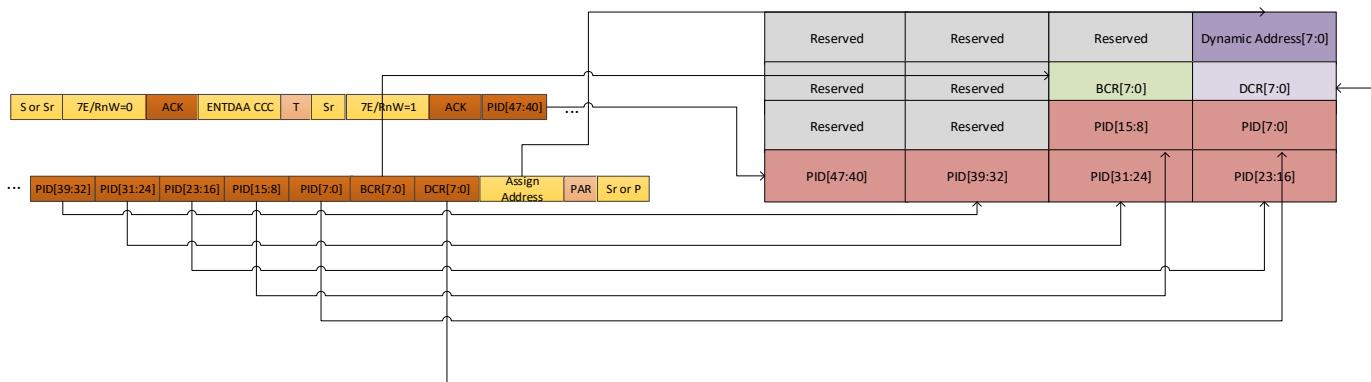
The controller does not distinguish read data coming from a private read transfer and read data coming from a CCC. The controller is also transparent to CCC, that is, it does not decode the CCC as a Master and does not treat CCCs differently from one another. The data packing logic for CCCs also follows the same rules as mentioned for private transfers. Therefore, for CCCs, it is expected that the software performs byte-swap while reading the data to get the required Byte endianness.

As an example, [Figure 2-25](#) and [Figure 2-26](#) show the byte-swapping to be performed by software for GETSTATUS CCC and GETPID CCC respectively.

Figure 2-25 Example 1: CCC - GETSTATUS (0x90)**Figure 2-26 Example 2: CCC - GETPID (0x8D)**

2.6.8.4.4 Packing of Received ENTDAA data by DWC_mipi_i3c into DCT Table.

The controller receives the bytes for the ENTDAA transfer in the same order as on the bus and places the first byte received in the Least Significant Byte position, and the following bytes in the LSB + 1 location, until all the four bytes of a word have been received in the DCT Table. The PID[47:16] is stored in the 'base' (DCT_SECTION_OFFSET[TABLE_OFFSET]) location and PID[15:0] is stored at 'base+1' location and BCR,DCR bytes at 'base+2' and Assigned dynamic address in 'base+3' location. This is shown in Figure 2-27.

Figure 2-27 Byte-Ordering - ENTDAA Transfer

2.6.9 SCL Generation and Timings Based on Bus Configuration

2.6.9.1 Overview of SCL Generation and Timings in DWC_mipi_i3c

MIPI I3C protocol supports three different bus configurations to allow the flexibility to the System Designer to determine the possible speed options. Bus configurations and their supported speed modes are shown in [Table 2-35](#).

Table 2-35 Bus Configuration and Supported Speed Modes

Speed Mode	Bus Configuration		
	Pure Bus (Only I3C Devices)	Mixed Fast Bus (Both I3C and I2C devices)	Mixed Slow and Limited Bus (Both I3C and I2C devices)
		Note: In this configuration, I2C Devices do not perform clock-stretching, and they have 50ns Spike filter.	Note: In this configuration, I2C Devices do not perform clock-stretching, and they do not have 50ns Spike filter.
SDR Mode Speed	Any speed to f_{SCL} (Max)	f_m , f_{m+} , f_{SCL_MIXED} (Min) to f_{SCL_MIXED} (Max), or slower using duty cycle	f_m or f_{m+} only
HDR-DDR Mode	Yes	Yes, using f_{SCL_MIXED} , or slower using duty cycle	No
HDR-TSP Mode	Yes	No	No
HDR-TSL Mode	Yes (but not needed)	Yes	No

As mentioned in [Table 2-35](#), the I3C Protocol Specification recommends to extend the low period (varying the duty-cycle), which changes the effective bus frequency in Pure Bus and Mixed Fast Bus system.

The High period should never exceed 45 ns (corresponding to a 11 MHz clock frequency), thus staying below the 50 ns required by the I2C Glitch/spike filter; however, the low period is free to be of any length permitted by the I3C device's allowed clock frequency range. The extended low period can be used for certain slow I3C devices supporting lower data rate (2, 4, 6, or 8 MHz).

The timing parameters for the subsequent transfer after the Address Header/Address Phase is selected based on the mode of the transfer:

- I3C Transfer: Push-Pull Timing
- I2C Transfer: Open-Drain Timing



- Note**
1. ACK for address phase of I3C transfer is always driven in Open-Drain mode.
 2. Complete ENTDAA transfer from START condition to STOP condition is generated in I3C Open-Drain mode.

2.6.9.2 DWC_mipi_i3c Timing Registers

The DWC_mipi_i3c controller supports the following registers to include the user-provided timing requirements of I3C protocol, based on the supported clock frequency:

- SCL I3C Open Drain Timing Register (SCL_I3C_OD_TIMING)
- SCL I3C Push Pull Timing Register (SCL_I3C_PP_TIMING)
- SCL I2C Fast Mode Timing Register (SCL_I2C_FM_TIMING)
- SCL I2C Fast Mode Plus Timing Register (SCL_I2C_FMP_TIMING)
- SCL Extended Low Count Timing Register (SCL_EXT_LCNT_TIMING)
- Bus Free and Available Timing Register (BUS_FREE_AVAIL_TIMING)
- SDA Hold Delay Timing Register (SDA_HOLD_DLY_TIMING)
- SCL Extended Termination Low count Timing Register (SCL_EXT_TERMIN_LCNT_TIMING)

2.6.9.2.1 SCL I3C Open Drain Timing Register

This register sets the SCL clock high period and low period count for I3C Open Drain transfers. The count value takes the number of core_clks to maintain the I/O SCL High/Low Period timing.

31	23	15	7	0
RESERVED	I3C_OD_HCNT	RESERVED	I3C_OD_LCNT	

- I3C_OD_LCNT: SCL Open-drain low count for I3C transfers targeted to I3C devices.
- I3C_OD_HCNT: SCL Open-drain high count for I3C transfers targeted to I3C devices.

2.6.9.2.2 SCL I3C Push Pull Timing Register

This register sets the SCL clock high period and low period count for I3C Push Pull transfers. The count value takes the number of core_clks to maintain the I/O SCL High/Low Period timing.

31	24 23	16 15	8 7	0
RESERVED	I3C_PP_HCNT	RESERVED	I3C_PP_LCNT	

- I3C_PP_LCNT: SCL Push-pull low count for I3C transfers targeted to I3C devices.
- I3C_PP_HCNT: SCL Push-pull high count for I3C transfers targeted to I3C devices.

2.6.9.2.3 SCL I2C Fast Mode Timing Register

This register sets the SCL clock high period and low period count for I2C Fast Mode transfers. The count value takes the number of core_clks to maintain the I/O SCL Low/High period timing.



- I2C_FM_LCNT: The SCL open-drain low count timing for I2C fast mode transfers.
- I2C_FM_HCNT: The SCL open-drain high count timing for I2C fast mode transfers.

2.6.9.2.4 SCL I2C Fast Mode Plus Timing Register

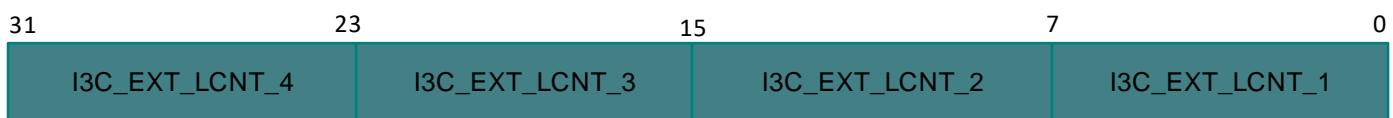
This register sets the SCL clock high period and low period count for I2C Fast Mode Plus transfers. The count value takes the number of core_clks to maintain the I/O SCL Low/High period timing.



- I2C_FMP_LCNT: The SCL open-drain low count timing for I2C fast mode plus transfers.
- I2C_FMP_HCNT: The SCL open-drain high count timing for I2C fast mode plus transfers.

2.6.9.2.5 SCL Extended Low Count Timing Register

This register sets the extended low periods for the I3C transfers to allow the low data rates of the slave devices.



- I3C_EXT_LCNT_*: The Extended Low counts to support the lower data rates as prescribed in GETMXDS CCC.
- The Speed field of Transfer command decides the selection of the respective low period to achieve the lower data rate for the transfers to Slave devices:
 - SDR1 - Uses I3C_EXT_LCNT_1 field for the data transfer.
 - SDR2 - Uses I3C_EXT_LCNT_2 field for the data transfer.
 - SDR3 - Uses I3C_EXT_LCNT_3 field for the data transfer.
 - SDR4 - Uses I3C_EXT_LCNT_4 field for the data transfer.

2.6.9.3 Bus Free and Available Timing Register

This Register sets the Bus free time for initiating the transfer in Master mode or generating IBI in Non-current Master mode.



- **BUS_FREE_TIME:** This field is used by the Master to initiate a new transfer after STOP condition. This field should be programmed equivalent to tCAS parameter.
- **BUS_AVAILABLE_TIME:** This field is used by the Slave to initiate a IBI after STOP condition.

2.6.9.3.1 SCL Extended Termination Low Count Timing Register

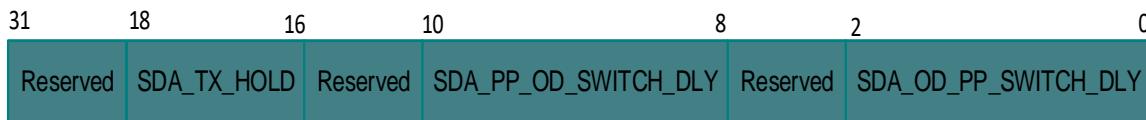
This register sets the extended SCL clock low period for Termination Bit of I3C read transfers and ternary skew count for HDR-TS transfers. The count value takes the number of core_clks to maintain the I/O SCL Termination Bit Low period timing. For more information, see the section "I3C SDR Read T-bit Phase Timings" in *DWC_mipi_i3c Controller User Guide*.



- **I3C_EXT_TERMN_LCNT:** The extended I3C Read Termination Bit low count for I3C Read transfers. This low count is used for extension of Termination Bit low period which can be different from I3C Push-pull low count.
- The effective Termination-Bit Low Period is derived based on the SDR speed as follows:
 - SDR0 speed: I3C_PP_LCNT + I3C_EXT_TERMN_LCNT
 - SDR1 speed: I3C_EXT_LCNT_1 + I3C_EXT_TERMN_LCNT
 - SDR2 speed: I3C_EXT_LCNT_2 + I3C_EXT_TERMN_LCNT
 - SDR3 speed: I3C_EXT_LCNT_3+ I3C_EXT_TERMN_LCNT
 - SDR4 speed: I3C_EXT_LCNT_4 + I3C_EXT_TERMN_LCNT
- **I3C_TS_SKEW_CNT:** Ternary Skew Count in terms of core_clks which is used for HDR Ternary Bus Turn-Around Detection in Master Mode.

2.6.9.3.2 SDA Hold Switch Delay Timing Register

This register sets the Data Hold time of SDA during transmit mode in SDR and HDR-DDR transfers. The count value takes the number of core_clks to maintain the I/O SDA Hold with respect to SCL timing.



- SDA_OD_PP_SWITCH_DLY: This field is used to delay the sda_out with respect to sda_oe while switching the transfer from OD1 (Open-Drain Mode SDA=1) to PP1 (Push-Pull Mode SDA=1). The valid value can range from 0 to 4. Others are reserved.
- SDA_PP_OD_SWITCH_DLY: This field is used to delay the sda_oe with respect to sda_out (in terms of the core_clock period) while switching the transfer from PP1 (Push-Pull Mode SDA=1) to OD1 (Open-Drain SDA=1). The valid value can range from 0 to 4. Others are Reserved.
- SDA_TX_HOLD: This field controls the hold time (in terms of the core_clock period) of the transmit data (SDA) with respect to the SCL edge in FM FM+ SDR and DDR speed mode of operations. This field is not applicable for the ternary speed modes. The valid values are 1 to 7. Others are Reserved.



- Note**
1. The SDA Hold is not applicable for HDR-TS transfers.
 2. SDA_OD_PP_SWITCH_DLY and SDA_PP_OD_SWITCH_DLY are applicable only in Single Master (IC_DEVICE_ROLE==1) configuration.

2.6.10 Derivation of I3C/I2C Timing Parameters from Timing Registers

This section describes how the timing parameters are derived from the Timing registers in different bus configurations.

Table 2-36 I3C Timing When Communication with I2C Legacy Devices

Parameter	Symbol	Legacy Mode 400KHz/FM	Legacy Mode 1MHz/FM+	Units	Notes
SCL Frequency	fSCL	$1/((I2C_FM_LCNT + I2C_FM_HCNT) * CCLK_PER)$	$1/((I2C_FMP_LCNT, I2C_FMP_HCNT) * CCLK_PER)$	MHz	
Setup Time for a Repeated START	tSU_STA	$I2C_FM_HCNT * CCLK_PER$	$I2C_FMP_HCNT * CCLK_PER$	ns	
Hold Time for a Repeated START	tHD_STA	$I2C_FM_HCNT * CCLK_PER$	$I2C_FMP_HCNT * CCLK_PER$	ns	
SCL Clock Low Period	tLOW	$I2C_FM_LCNT * CCLK_PER$	$I2C_FMP_LCNT * CCLK_PER$	ns	
SCL Clock High Period	tHIGH	$I2C_FM_HCNT * CCLK_PER$	$I2C_FMP_HCNT * CCLK_PER$	ns	
Data Setup Time	tSU_DAT	$(I2C_FM_LCNT - SDA_TX_HOLD) * CCLK_PER$	$(I2C_FMP_LCNT - SDA_TX_HOLD) * CCLK_PER$	ns	Data Setup is considered for the write transfer from Controller alone point of view. The system delays are not considered.

Table 2-36 I3C Timing When Communication with I2C Legacy Devices

Parameter	Symbol	Legacy Mode 400KHz/FM	Legacy Mode 1MHz/FM+	Units	Notes
Data Hold Time	tHD_DAT	SDA_TX_HOLD*CCLK_K_PER	SDA_TX_HOLD*CCLK_PER	ns	Data Hold is considered for the write transfer from Controller alone point of view. The system delays are not considered.
Setup Time for STOP	tSU_STO	I2C_FM_HCNT*CCLK_K_PER	I2C_FMP_HCNT*CCLK_PER	ns	
Bus Free Time Between a STOP Condition and a START Condition	tBUF	I3C_MST_FREE*CCLK_K_PER	I3C_MST_FREE*CCLK_PER	ns	
Pulse width of Spikes that the Spike Filter Must Suppress	tSPIKE	NA	NA		Spike filter is not included in the DWC_mipi_i3c

Table 2-37 I3C Open Drain Timing Parameters

Parameter	Symbol	I3C Open Drain Mode	Units	Notes
Low Period of SCL Clock	tLOW_OD	I3C_OD_LCNT*CCLK_PER	ns	
High Period of SCL Clock	tHIGH	I3C_OD_HCNT*CCLK_PER	ns	
SDA Data Setup Time During Open Drain Mode	tSU_OD	(I3_OD_LCNT – SDA_TX_HOLD)*CCLK_PER	ns	
Clock After START Condition	tCAS	I3C_MST_FREE*CCLK_PER	ns	
Clock Before STOP Condition	tCBP	I3C_MST_FREE*CCLK_PER	ns	
Current Master to Secondary Master Overlap time during handoff	tMMOverlap	I3C_OD_LCNT*CCLK_PER	ns	This parameter is for secondary Master and not applicable in Single Master system.

Table 2-37 I3C Open Drain Timing Parameters

Parameter	Symbol	I3C Open Drain Mode	Units	Notes
Bus Available Condition	tAVAIL	NA	ns	This parameter is used to generate IBI and is not applicable in Master Mode.
Bus Idle Condition	tIDLE	NA	ns	This parameter is used to generate HJ and is not applicable in Master Mode.

Table 2-38 I3C Push-Pull Timing Parameters for SDR and HDR-DDR Modes

Parameter	Symbol	I3C Push-Pull Mode	Units	Notes
SCL Clock Frequency	fSCL	$1/((I3C_PP_LCNT+I3C_PP_HCNT)*CCLK_PER)$	MHz	
SCL Clock Low Period	tLOW	$I3C_PP_LCNT*CCLK_PER$	ns	For HDR-DDR Mode, the 'I3C_PP_HCNT' register is considered for Low period. This is because the Legacy I2C devices should not see the transitions of DDR and TSP Modes and it should be filtered out with spike filters in I2C
SCL Clock High Period (Mixed and Pure Bus)	tHIGH	$I3C_PP_HCNT*CCLK_PER$	ns	
Clock in to Data out for Slave	tSCO	NA	ns	This Parameter is for Slave to drive data w.r.t sampling edge of SCL and is not applicable in Master Mode.
SDA Signal Data Hold in Push-Pull Mode	tHD_PP	$SDA_TX_HOLD*CCLK_PER$	ns	Data Hold is considered for the write transfer from Controller alone point of view. The system delays are not considered.
SDA Signal Data Setup in Push-Pull Mode	tSU_PP	$(I3C_PP_LCNT-SDA_TX_HOLD)*CCLK_PER$	ns	Data Setup is considered for the write transfer from Controller alone point of view. The system delays are not considered.
Clock After Repeated START	tCASr	$I3C_MST_FREE*CCLK_PER$	ns	
Clock Before Repeated START	tCBSr	$I3C_MST_FREE*CCLK_PER$	ns	
Capacitive Load per Bus Line (SDA/SCL)	C _b	NA	ns	This Parameter is system level parameter and not applicable for DWC_mipi_i3c Component.

Table 2-39 I3C Push-Pull Timing Parameters for HDR-TSP and HDR-TSL Modes

Parameter	Symbol	I3C Push-Pull Mode (HDR-TSP and HDR-TSL)	Units	Notes
Edge-to-Edge Period	tEDGE	I3C_PP_HCNT*CCLK_PER	MHz	
Allowed Difference Between Signals for 'Simultaneous' Change	tSKEW	NA	ns	DWC_mipi_i3c controller does not generate any skew between the SCL and SDA lines.
Stable Condition Between Symbols	tEYE	I3C_PP_HCNT*CCLK_PER	ns	
Time Between Successive Symbols	tSYMBOL	I3C_PP_HCNT*CCLK_PER	ns	

2.6.11 Error Detection

This section describes the error detection methods in DWC_mipi_i3c.

2.6.11.1 Detecting Error Type in the Processed Commands

The ERR_STATUS field in the RESPONSE_QUEUE_PORT register is used to detect the error type in the processed commands.

This field indicates errors based on the codes shown in [Table 2-34](#).

Table 2-40 Error Codes

ERROR_CODE	ERROR_TYPE	ERROR_DESCRIPTION
0x0	NO_ERR	Initiated transfer is successful without any errors
0x1	CRC_ERR	CRC Error occurred in the HDR-DDR Read Transfer
0x2	PARITY_ERR	Parity Error occurred in HDR-DDR or HDR-TS Read Transfers
0x3	FRAME_ERR	Frame Error occurred in HDR-TS Read Transfer
0x4	ADDR_HDR_NACK_ERR	Received NACK for the Address Header. Indicates no I3C Slave is present in the system
0x5	ADDR_NACK_ERR	Received NACK for Slave Address of Write/Read Transfer Or Received NACK for Assign Address of ENTDAA Transfer

Table 2-40 Error Codes

ERROR_CODE	ERROR_TYPE	ERROR_DESCRIPTION
0x6	OVL_URL_ERROR	Experienced Receive FIFO OverFlow in HDR Transfers Or Experienced Transmit FIFO UnderFlow in HDR Transfers
0x7	Reserved	Reserved
0x8	ABORT_ERR	Transfer is Aborted based on the User initiated Abort
0x9	I2C_WR_DATA_NACK_ERR	Received NACK for the I2C Write Data Transfer

2.6.11.2 SDR Error Detection and Recovery Methods

Table 2-41 lists the two error types for SDR transfers and their recovery methods.

Table 2-41 Error Types

Error Type	Description	Error Detection Method	Error Recovery Method
M0	Transaction after sending CCC	Detects illegally formatted CCC	Stop the transmission, then send STOP and retry the transmission
M2	No response to Broadcast Address (7'h7E)	Master detects NACK after Broadcast Address (7'h7E) transmission	Upon detection of NACK, Master transmits HDR Exit Pattern followed by STOP

2.6.11.2.1 Error Type M0

If the Master detects an illegally formatted CCC, then the Master stops the transmission, send STOP, and retry the transmission.

An example of an illegally formatted CCC is the Master receiving just one byte from the Slave in a GETMWL CCC code. This is because the Master expects two bytes.

DWC_mipi_i3c does not decode the CCC as it is transparent for the application. The controller always initiates the CCC transfer and expects the number of bytes as mentioned in DATA_LENGTH field of Transfer Argument.

If the slave terminates earlier than the DATA_LENGTH expected by the controller, the controller represents it in the DATA_LENGTH of the RESPONSE.

The application has to decode this information through RESPONSE of the CCC transfer and can re-issue the same command in the COMMAND_QUEUE for re-transmission.

2.6.11.2.2 Error Type M2

If the Master does not receive an ACK of a transmitted Broadcast Address (7'h7E), then it transmits the HDR Exit Pattern followed by STOP.

DWC_mipi_i3c controller always generate STOP upon receiving NACK for the Broadcast Address(7'h7E) and enters the HALT state. This can be detected through the "TRANSFER_ERR_STAT" interrupt in INTR_STAT register.

2.6.12 Defining Byte Support

The Defining Byte Support feature is an extended feature to the existing CCC transfers model. The Defining Byte is included after the CCC Code and before the payload (in broadcast CCC's)/Restart (Directed CCC's) to indicate an extended capability or an action associated with the CCC. The Defining Byte is supported for both Broadcast and Directed SDR CCC transfers. In both cases, the Defining Byte is included after CCC Code byte as sub command before the Payload. [Figure 2-28](#) and [Figure 2-29](#) describe the placement of Defining Byte in Broadcast CCC and Directed CCC transfers respectively.

Figure 2-28 Broadcast CCC with Defining Byte

START S (Sr)	Address Header (0x7E,w)	Broadcast CCC Code (CCC,T)	Defining Byte (DF_Byte,T)	Data Payload (DB,T)	STOP (P/Sr)
-----------------	----------------------------	-------------------------------	------------------------------	------------------------	----------------

Figure 2-29 Directed CCC with Defining Byte

START (S/Sr)	Address Header (0x7E,w)	Directed CCC Code (CCC,T)	Defining Byte (DF_Byte,T)	RESTART (Sr)	...
...	Slave Address (SA,w/r)	Data Payload (DB,T)	STOP (P/Sr)		

The Defining Byte is enabled for Broadcast or Directed SDR CCC transfers by setting the 'Defining Byte Present (DBP)' bit in Transfer Command and the Defining Byte value is captured from 'Defining Byte (DB)' field of Transfer Argument or Short Data Argument.

2.6.13 Packet Error Check

The Packet Error check feature enables the generation and validation of 8-bit CRC on the data stream between Master and Slave for reliable communication. The Transmitter always inserts an extra PEC byte at the end of the data stream and receiver validates it. The PEC is applicable for SDR CCC, IBI, and SDR Private Transfers. The Host or device implements an 8-bit Packet Error Code (PEC), which is appended at the end of all transactions if PECs are enabled through DEVCTRL CCC (JEDEC Sideband Specification). The PEC is a CRC-8 value calculated on all the messages bytes except for START, REPEATED START, STOP conditions or T-bits, ACK and NACK, and IBI header (7'h7E followed by W=0) bits. The polynomial for CRC-8 calculation is 'X8 + X2 + X1 + 1'. The PEC byte is supported only for SDR Transfers and not applicable for HDR Transfers.

The following figures depict the inclusion of PEC byte for different types of transfers.

Figure 2-30 Broadcast CCC with PEC Byte

START S (Sr)	Address Header (0x7E,w)	Broadcast CCC Code (CCC,T)	Data Payload (DP,T)	PEC Byte (PEC,T)	STOP (P/Sr)
---------------------	--------------------------------	-----------------------------------	----------------------------	-------------------------	--------------------

Figure 2-31 Directed CCC with PEC Byte

START S (Sr)	Address Header (0x7E,w)	Broadcast CCC Code (CCC,T)	PEC Byte (PEC,T)	RESTART (Sr)
	Slave Address (slv addr,w)	Data Payload (DP,T)	PEC Byte (PEC,T)	STOP (P/Sr)

Figure 2-32 Private Write Transfer with PEC Byte

START S (Sr)	Slave Address (slv addr,w)	Data Payload (DP,T)	PEC Byte (PEC,T)	STOP (P/Sr)
---------------------	-----------------------------------	----------------------------	-------------------------	--------------------

Figure 2-33 Private Read Transfer with PEC Byte

START S (Sr)	Slave Address (slv addr,r)	Data Payload (DP,T)	PEC Byte (PEC,T)	STOP (P/Sr)
---------------------	-----------------------------------	----------------------------	-------------------------	--------------------

The Packet Error check can be enabled for SDR Transfers through setting ‘PEC’ bit of Transfer Command to ‘1’. When this bit is set, the master controller generates PEC byte at the end of Write transfer and validates the PEC byte at the end of Read transfer. If the received PEC byte for the Read transfer does not matches the internally generated PEC byte based on the data received from the slave, then Master controller generates ‘Transfer Error’ interrupt and indicates it as ‘PEC Error’ in Response Status. The Master controller enables the Packet Error checking for IBI with Data transfers if DAT[IBI_PEC_EN] bit is set to 1.

2.6.14 Broadcast CCC’s in I2C Speed

JEDEC Sideband Specification allows to issue certain broadcast CCC’s (DEVCTRL and SETHID) in I2C Mode to indicate Bus context before Address assignment with SETAASA CCC. The JEDEC Sideband specification mandates to send these broadcast CCC’s from the Master in limited speed 1MHz to guarantee that these CCC’s are registered by the slave devices without any error.

You can issue Broadcast CCC’s with I2C FM speed by setting the TRANSFER_COMMAND[SPEED] field to ‘I2C FM’. The Master Controller generates those CCC’s in I3C protocol mode with I2C FM speed.

2.6.15 BUS RESET Generation

To prevent a malfunctioning device from locking up the I3C/Sideband bus, I3C protocol and JEDEC Sideband Specifications defines a bus protocol RESET mechanism.

The Master Controller supports the generation of following two patterns:

- Timed Reset
- HDR-EXIT Pattern

The Reset Pattern is selected in the 'RESET_CONTROL.BUS_RESET_TYPE' field and to request the command is required to be written to 1'b1 to 'RESET_CONTROL.BUS_RESET' register field. The DWC_mipi_i3c controller considers this request only when it is in IDLE state and gives priority over scheduled regular commands. The Master controller generates the Timed Reset by driving SCL Low and SDA low for "SCL_LOW_MST_TIMEOUT[SCL_LOW_MST_TIMEOUT_COUNT]" period of time. Once the controller completes the Timed Reset, the controller auto-clears the 'RESET_CONTROL.BUS_RESET' request bit. The software can poll this request bit to know whether the controller completes the generation of the Requested Reset pattern, an interrupt is generated representing that the Request Reset pattern is issued.

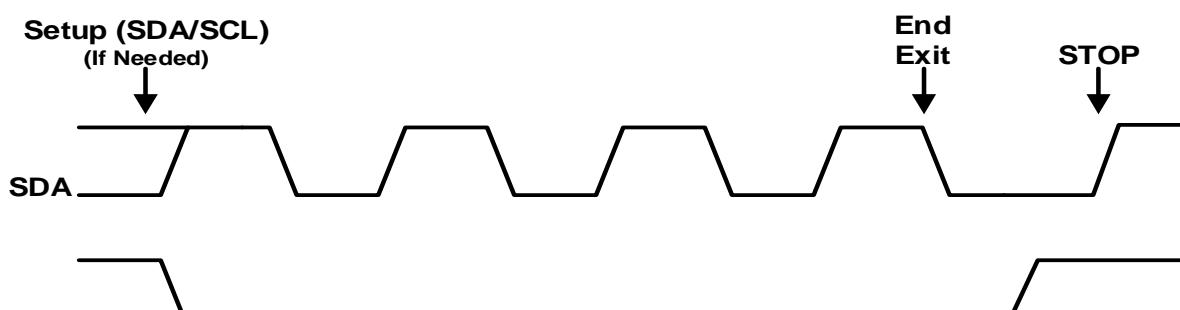
[Figure 2-34](#) and [Figure 2-35](#) show the SCL Timed Reset and HDR-EXIT Pattern respectively.

Figure 2-34 SCL Timed Reset



2'b11 : SCL Low Timeout Reset

Figure 2-35 HDR Exit Reset pattern



2'b00 : HDR Exit Pattern Plus STOP

2.7 Slave Functionality with DWC_mipi_i3c

2.7.1 Overview of Slave Role in DWC_mipi_i3c

The Dual-Role DWC_mipi_i3c can be configured either as a Slave only controller or as a secondary Master. The following slave functions are applicable when the controller is operating as a Slave.

2.7.2 Description of the Slave Role in DWC_mipi_i3c

The DWC_mipi_i3c Slave controller can be selected either as a static address device or a dynamic address only device.

This selection can be done in the following ways:

- Driving the strap input, static_addr_en and static_addr:
 - static_addr_en=1 implies that the controller is a static address device.
 - static_addr_en=0 implies that the controller is a dynamic address only device.
- Programming the DEVICE_ADDR[STATIC_ADDR_VALID] and DEVICE_ADDR[STATIC_ADDR] register.
 - The reset value of these fields are set from strap input static_addr_en and static_addr. The application can overwrite this value by programming this register before enabling the controller.

If the controller is selected as a static address device, then the device responds for both ENTDAA and SETDASA CCC commands from the Current Master until the dynamic address is assigned successfully.

If the controller is selected as a dynamic address device, then the device responds for only ENTDAA CCC command from the Current Master until the dynamic address is assigned successfully.

Once the dynamic address is assigned and valid, then the controller stops responding for the ENTDAA and SETDASA CCC commands until the Dynamic address is reset through RSTDAA CCC command.

The SETNEWDA CCC from the Current Master allows the slave controller to replace the current Dynamic Address with a new Dynamic Address.

2.7.3 I3C vs I2C Role Selection

The DWC_mipi_i3c Slave controller can be selected as an I3C slave or as an I2C slave by two ways:

- If the input pin I2C_mode is tied to 1, it brings up the Slave controller as an I2C slave. Once selected as an I2C slave, the role cannot be changed until reset and after the input pin I2C_mode is made zero.
- Adaptive Selection: The adaptive selection is possible only when the input pin “mode_i2c” is tied to 0. During initial power-up, the DWC_mipi_i3c controller is in I2C mode as it does not have any Dynamic Address (DA) assigned to it. The mode changes to I3C only when a DA is assigned by any of Dynamic Address assignment procedure given in MIPI I3C Specification. The mode again changes back to I2C when RSTDAA CCC is issued or on hardware reset.

To enable the static address match to happen, the application must enable the static address as mentioned in the section “[Description of the Slave Role in DWC_mipi_i3c](#)” on page [135](#). Once in I2C role, the Slave controller ignores all I3C related transfers like IBI, Hot-Join and all the CCCs.

In I2C mode of operation, the clock stretching is not supported by Slave controller. The spike filter required for I2C device should be present outside the controller interface and is not a part of the deliverable, only the control to enable or disable the spike filter is provided from the controller. Until the Dynamic Address is assigned either through ENTDAA or SETDASA CCC, the controller remains in I2C mode. Once the Dynamic Address is assigned and valid, then the controller switches to I3C mode.

If the auto generation of Hot-Join is to be disabled until the correct mode of operation is identified, then the slave application must set the ADAPTIVE_I2C_I3C bit in the DEVICE_CTRL register. If this bit is set, then the Slave controller initiates a Hot-Join request to the I3C Master only when a 7E header is received on the bus and changes to I3C mode of operation.

2.7.3.1 Conditions for Mode Change

On power-up, the Slave controller is in I2C mode and it changes state under the following conditions:

- The Slave controller changes to I3C mode when a dynamic address is assigned by any of the Dynamic Address assignment procedures given in I3C specification.
- The Slave controller changes to I2C mode when a RSTDAA CCC is received from the Master. The Slave controller responds to transfers matching its static address in I2C mode.
- On power-on, reset, the controller is in I2C mode.

2.7.3.2 Conditions for Glitch Filter Enable/Disable

On power-up, the glitch filter is enabled. In adaptive mode, the glitch filter is disabled by the controller when the first 7E header is received. Once disabled, it is in disable state until Power On Reset. The glitch filter is always enabled if input pin mode_i2c is tied to 1.

2.7.4 Slave Role Related Registers

The following are the Slave Role specific registers/fields in DWC_mipi_i3c (these registers are also explained in [Chapter 5, “Register Descriptions”](#)):

- IDLE_CNT_MULTIPLIER (under DEVICE_CTRL register)
- STATIC_ADDR_VALID and STATIC_ADDR (under DEVICE_ADDR register)
- DYN_ADDR_ASSGN_STAT (under INTR_STATUS register)
- SLV_PID_VALUE
- SLV_CHAR_CTRL
- SLV_MAX_LEN
- MAX_READ_TURNAROUND
- MAX_DATA_SPEED
- PRESENT_STATE
- SLV_EVENT_STATUS
- CCC_DEVICE_STATUS
- SLV_INTR_REQ

2.7.5 Handling Address Assignment

The configured Slave controller can either be selected as a static address device or a dynamic address device. The static address can be configured as mentioned in the section “[Description of the Slave Role in DWC_mipi_i3c](#)” on page 135.

If the controller is selected as a static address device, then the device responds for both ENTDAA and SETDASA CCC commands from the Current Master until the dynamic address is assigned successfully.

If the controller is selected as a dynamic address device, then the device responds for only ENTDAA CCC command from the Current Master until the dynamic address is assigned successfully.

Once the dynamic address is assigned and valid, then the controller stops responding for the ENTDAA and SETDASA CCC commands until the Dynamic address is reset through RSTDAA CCC command.

2.7.6 CCC Transfers with DWC_mipi_i3c Slave

2.7.6.1 Overview of CCC Transfers in Slave Mode of Operation

The CCC Transfers shown in [Table 2-42](#) are supported when DWC_mipi_i3c is operating in Slave mode. All unsupported CCC commands are ignored (NACKED) by the slave controller.

Table 2-42 Supported CCC Transfers

CCC	Type
ENTDAA	Broadcast
SETDASA	Directed
GETSTATUS	Directed
GETMXDS	Directed
ENTHDR0	Broadcast
ENTHDR1	Broadcast
ENTHDR2	Broadcast
GETMRL	Directed
SETMRL	Broadcast, Directed
GETMWL	Directed
SETMWL	Broadcast, Directed
ENECA	Broadcast, Directed
DISEC	Broadcast, Directed
RSTDAA	Broadcast, Directed
SETNEWDA	Directed
GETPID	Directed

Table 2-42 Supported CCC Transfers (Continued)

CCC	Type
GETBCR	Directed
GETHDRCAP	Directed
ENTAS0	Broadcast, Directed
ENTAS1	Broadcast, Directed
ENTAS2	Broadcast, Directed
ENTAS3	Broadcast, Directed
DEFSLVS	Broadcast
GETACCMST	Directed

All the CCCs in [Table 2-42](#) are handled within the Slave controller without involving the slave application. The CCC write data from the Current Master is either captured in a register or consumed within the controller. The CCC read data from the controller is sourced either from the configured parameters or the registers/port inputs maintained within the controller.

Optionally, the Slave controller (if enabled) can generate a common interrupt (CCC_UPDATED_STAT interrupt in INTR_STATUS register) when the Current Master updates any of the following register values through a CCC transfer:

- Dynamic Address Assignment through ENTDAA/SETDASA (including RSTDAA, SETNEWDA, Assign new DA)
- Maximum Read Length
- Maximum Write Length
- Enable/Disable Slave events command ENEC, DISEC
- Enter activity state ENTAS0/1/2/3

2.7.6.2 Description of CCC Transfers with DWC_mipi_i3c Slave

The DWC_mipi_i3c Slave controller collects the required CCC information either through configurable parameters or from input port signals, to enable the controller to handle the CCC transfers. During run time, the controller does not involve the application to handle any CCC transfers. Upon receiving a CCC command from the I3C Master, the controller handles it in one of the three ways — through configurable parameters, through port signals or through internal registers.

2.7.6.2.1 CCCs Handled through Configuration Parameters

Some CCC related parameters set while configuring the controller are used to return Slave capabilities when queried by the Master through Read CCCs. Most of the capabilities listed in [Table 2-43](#) are controlled only by configuration parameter and cannot be modified. However, the value of a few capabilities such as IC_SLV_HDR and IC_SLV_DATA_SPEED_LIMIT can be overwritten by register (SLV_CHAR_CTRL) programming.

Table 2-43 CCS Handled through Configuration Parameters

CCC	Related Parameters
GETBCR	IC_SLV_HDR, IC_SLV_BRIDGE, IC_SLV_OFFLINE_CAP, IC_SLV_IBI, IC_SLV_IBI_DATA, IC_SLV_DATA_SPEED_LIMIT, and SLV_CHAR_CTRL register
GETHDRCAP	IC_SLV_HDR_DDR, IC_SLV_HDR_TSP, IC_SLV_HDR_TSL

2.7.6.2.2 CCCs Handled through Port Signals

The CCCs that are related to dynamic functionality of the controller are handled through port signals.

Unlike CCCs handled through configuration parameters, these CCCs can be changed during runtime. The data associated with write CCC is made available to the application through output ports and data associated with read CCCs is collected through input ports. [Table 2-44](#) lists the CCCs handled through port signals.

Table 2-44 CCCs Handled through Port Signals

CCC	Related Port Signal
ENTAS0	act_state
ENTAS1	act_state
ENTAS2	act_state
ENTAS3	act_state
GETSTATUS	act_mode, pending int

2.7.6.2.3 CCCs Handled through Internal Registers

Some CCCs are handled internally to the controller, with no or minimal user inputs. [Table 2-45](#) lists the CCCs that are handled through internal registers.

Table 2-45 CCCs Handled through Internal Registers

CCC	Related Internal Register
ENECA	Updates the SLV_EVENT_STATUS register and automatically controls the generation of MR, SIR and HJ requests from the controller.
DISEC	Updates the SLV_EVENT_STATUS register and automatically controls the generation of MR, SIR and HJ requests from the controller.
RSTDAA	Clears DEVICE_ADDR.[DYNAMIC_ADDR] and DEVICE_ADDR.[DYNAMIC_ADDR_VALID].
ENTHDR0	Controller starts operating in HDR_DDR mode.
ENTHDR1	Controller starts operating in HDR_TSP mode.
ENTHDR2	Controller starts operating in HDR_TSL mode.
SETDASA	Upon receiving SETDASA CCC, if static address matches with incoming address, DEVICE_ADDR.[DYNAMIC_ADDR] and DEVICE_ADDR.[DYNAMIC_ADDR_VALID] are updated.
SETNEWDA	Updates DEVICE_ADDR.[DYNAMIC_ADDR].
GETMWL	Transmits the value from the SLV_MAX_LEN.MWL register to the I3C Master.
GETMRL	Transmits the value from the SLV_MAX_LEN.MRL register to the I3C Master.
SETMWL	Updates the value of SLV_MAX_LEN.MWL register after receiving the SETMWL CCC.
SETMRL	Updates the value of SLV_MAX_LEN.MRL register after receiving the SETMRL CCC.

2.7.6.2.4 CCCs Handled Through Both Ports and Registers

To support some CCCs, you must provide inputs. The controller supports these CCCs using programmable registers, which you can modify. The reset value of the register is driven from input ports provided. So, if you do not want to modify the data related to CCC, you can drive the relevant data in the corresponding

input port. To change the data driven through input ports, you can program the corresponding register with relevant data before enabling the controller.

Table 2-46 CCCs Handled through Ports and Registers

CCC	Related Ports and Register
ENTDAA	All input ports, registers and parameters related to GETPID, GETBCR and GETDCR
GETPID	slv_pid[47:0] input port SLV_MIPI_ID_VALUE register SLV_PID_VALUE register
GETDCR	slv_dcr[7:0] input port DCR field in SLV_CHAR_CTRL register

2.7.6.3 Handling of GETMXDS CCC

Whether or not GETMXDS CCC is ACK'ed by the Slave Controller is determined by the value programmed in SLV_CHAR_CTRL[MAX_DATA_SPEED_LIMIT]. Data for GETMXDS CCC is sourced in two different ways (either from Parameters or from Programmable Registers) unlike other CCCs that are solely serviced only from one of the following sources, that is, from Parameters /Ports/Registers.

This is controlled by the configuration parameter IC_SLV_MXDS_PROG.

- IC_SLV_MXDS_PROG not set:
 - The value returned by the slave for GETMXDS CCC is sourced from the configuration parameters that are provided while configuring the controller.
GETMXDS: IC_SLV_MXDS_MAX_WR_SPEED, IC_SLV_MXDS_MAX_RD_SPEED,
IC_SLV_MXDS_CLK_DATA_TURN, IC_SLV_MXDS_MAX_RD_TURN
- IC_SLV_MXDS_PROG is set:
 - Some of the values returned by the slave for GETMXDS CCC is sourced from the following parameters that are provided while configuring the controller.
GETMXDS: IC_SLV_MXDS_MAX_RD_TURN
 - Values for "Clock to Data Turnaround Time (Tsc0)" and "Maximum Sustained Data Rate for non-CCC messages sent by Slave Device to Master Device" and "Maximum Sustained Data Rate for non-CCC messages sent by Master Device to Slave Device" can be sourced either through input straps or through register programming as outlined:
 - Driving the strap input, slv_clk_data_turn_time, slv_max_rd_speed, and slv_max_wr_speed.
 - Programming the MAX_DATA_SPEED.[MXDS_CLK_DATA_TURN],
MAX_DATA_SPEED.[MXDS_MAX_RD_SPEED], and
MAX_DATA_SPEED.[MXDS_MAX_WR_SPEED].
 The reset value of these fields are set from strap input slv_clk_data_turn_time, slv_max_rd_speed, and slv_max_wr_speed. The application can overwrite this value by programming this register before enabling the controller.



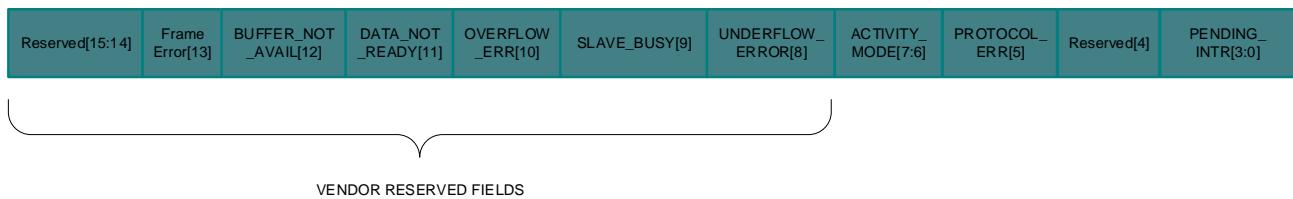
Note It is recommended to select the configuration parameter IC_SLV_MXDS_PROG at the time of RTL configuration. This allows the flexibility of providing timing related information related to GETMXDS through programming after post silicon characterization.

If the slave is known to have max data speed limitation, reserved values of maxWr, maxRd and tSCO (through Parameters/Strap/Register) can be used to establish a private contract between master and slave to communicate the speed limitation.

2.7.6.4 Handling of the GETSTATUS CCC

The format for the GETSTATUS CCC as supported by DWC_mipi_i3c is given as follows.

Figure 2-36 Handling of the GETSTATUS CCC



The GETSTATUS vendor reserved fields are used by DWC_mipi_i3c controller to indicate to the master the error conditions that has occurred in the previous transfer. In case of any error, the DWC_mipi_i3c controller in slave mode of operation, NACKS the next private transfer coming from the Master. The Master can understand the status of the transfer which was not successfully accepted or transmitted by the DWC_mipi_i3c controller by issuing the GETSTATUS CCC.

The CCC_DEVICE_STATUS register reflects all the information passed to the master when Master issues GETSTATUS CCC.

A detailed description of the error fields and other status fields present in the GETSTATUS CCC format supported by DWC_mipi_i3c controller in slave mode is provided in the description of CCC_DEVICE_STATUS register.

2.7.6.5 Handling of ENTDA, GETPID and GETDCR

It might not always be possible to determine the PID and DCR values at the time of RTL configuration. The configuration parameter IC_SLV_UNIQUE_ID_PROG makes it possible to provide these values at run time by programmable registers or straps as opposed to configuring the values during RTL generation.

When IC_SLV_UNIQUE_ID_PROG is set to 0

- Of the 48 bits of the Provisional ID, 44 bits are configured during RTL generation and only Instance ID (Bits[15:12]) is changeable at run time. Instance ID can be programmed using inst_id input port and SLV_PID_VALUE.[SLV_INST_ID] register. The remaining 44 bits of PID value is obtained by the controller from the following Configuration Parameters:
 - ❑ IC_SLV_MIPI_MFG_ID
 - ❑ IC_SLV_PROV_ID_SEL
 - ❑ IC_SLV_PART_ID

- IC_SLV_PID_DCR.
- The value of DCR (Device Characteristic Register) is also configured during RTL generation using the Configuration Parameter IC_SLV_DCR_VALUE.

When IC_SLV_UNIQUE_ID_PROG is set to 1

- This makes it possible for values of Provisional ID and DCR to be provided by driving the straps and programming the registers. The value of 48-bit Provisional ID can be provided through straps by driving strap input slv_pid[47:0] and also by programming the registers SLV_MIPI_ID_VALUE and SLV_PID_VALUE registers.

The value of DCR can be provided through straps by driving strap input slv_dcr[7:0] and also by programming DCR field in SLV_CHAR_CTRL register. Detailed description of the strap inputs and register fields provided for programming the PID and DCR values is as follows:

- PID
 - PID[47:33]: MIPI Manufacturer ID
Sourced from slv_pid[47:33] input port and SLV_MIPI_ID_VALUE.[SLV_MIPI_MFG_ID] register
 - PID[32]: Provisional ID Type Selector
Sourced from slv_pid[32] input port and SLV_MIPI_ID_VALUE.[SLV_PROV_ID_SEL] register
 - PID[31:16]: Part ID
Sourced from slv_pid[31:16] input port and SLV_PID_VALUE.[SLV_PART_ID] register
 - PID[15:12]: Instance ID
Sourced from slv_pid[15:12] input port and SLV_PID_VALUE.[SLV_INST_ID] register
 - PID[11:0] Additional meaning
Sourced from slv_pid[11:0] input port and SLV_PID_VALUE.[SLV_PID_DCR] register
- DCR
Sourced from slv_dcr input port and SLV_CHAR_CTRL.[DCR] register

The reset value of these register fields are set from strap inputs. The application can override this value by programming this register before enabling the controller

2.7.6.6 CCC Transfer Related Registers

The following are the registers used by CCC transfers either to source data for responding to incoming CCCs or to update the values set by incoming CCC. These registers are also explained in [Chapter 5, "Register Descriptions"](#).

- CCC_UPDATED_STAT
- CCC_UPDATED_STAT_EN
- DEVICE_ADDR
- SLV_PID_VALUE
- SLV_MIPI_ID_VALUE
- SLV_CHAR_CTRL

- SLV_EVENT_STATUS
- SLV_MAX_LEN
- MAX_READ_TURNAROUND
- MAX_DATA_SPEED

2.7.7 Private Data Transfers

2.7.7.1 Overview of Private Data Transfers

The DWC_mipi_i3c Slave controller supports private read and write transfers through I3C interface over the SCL and SDA lines.

For Private Receive and Transmit Transfers in SDR mode of operation, if the Slave controller can accept or transmit the transaction from the Master, it accepts the address by providing an ACK response else the address is NACKED.

In HDR Mode if the Slave controller is not able to accept the Private Write transaction from the Master, the data is dropped because of no mechanism to NACK private Write Transfer. For Private Read transfer a NACK response (for DDR) or an exit pattern (for TSP/TSL) is issued.

Receiving a NACK means that an error exists in the current or previous transfer Read/Write, and the Master can choose to do one of the following:

- Issue a Start/Restart and re-attempt the transfer. If the previous transfer is NACKed because of unavailability of internal buffer space, the second transfer might be successful.
- Issue a GETSTATUS CCC to check if the NACK is due to a more permanent condition encountered by the previous transfer due to some protocol error like parity/crc or due to overflow/underflow errors and then re-issue the NACK transfer.

2.7.7.2 Handling Private Receive (Master Write) Transfers

In SDR mode of operation, the Slave controller accepts the write address (ACK response) when all of the following conditions are met:

- Receive Buffer has the space (in terms of empty locations) equal or more than the programmed value of the RX_START_THLD register.
- Response Queue is not full and it has some space to hold the response for the current transfer.

Otherwise, the controller responds with a NACK for the private write address.

Once the write address is acknowledged with ACK (accepted), the controller expects the RX FIFO space to be available until the end of transfer to avoid overflow condition.

In the PIO mode (non-DMA) of operation, use either the RX_BUF_THLD interrupt to free up the RX FIFO space while the RX data being received, or configure the RX FIFO to accommodate the entire write transfer data (when the maximum write transfer size is defined and small).

In the DMA mode of operation, the DMA request of the handshake interface is initiated when the RX FIFO level reaches the programmed RX_BUF_THLD level.

When an overflow condition is encountered (when the system latency is high to consume the receiving data), the controller sets the CCC_DEVICE_STATUS [OVERFLOW_ERR] bit of the register and drops further incoming data until the termination (either STOP or RESTART) is detected.

When a parity/CRC error is encountered during the Master Write transfer, the controller sets the CCC_DEVICE_STATUS [PROTOCOL_ERROR] bit of the register and drops further incoming data until the termination is detected.

The controller, once it sets the OVERFLOW_ERROR or PROTOCOL_ERROR bit, rejects (NACK) any further private transfer request from the Master until the Master reads the device status through a GETSTATUS CCC and the slave application resumes the slave operation by setting the bit DEVICE_CTRL [RESUME].

If the receive FIFO does not have threshold amount of space to accommodate the write transfer, the controller NACKs the request and the CCC_DEVICE_STATUS[BUFFER_NOT_AVAIL] bit is set. The controller, once it sets the BUFFER_NOT_AVAIL bit, rejects (NACK) any further private write transfer request from Master until space is available in the receive FIFO.

For more information on error handling mechanism, see the "Error Handling" section.



Note In I2C mode of operation, if an overflow error is encountered in a write transfer, the controller terminates the ongoing transfer by sending a NACK. The controller rejects (NACK) any further private transfer request from Master until the slave application resumes the slave operation by setting the DEVICE_CTRL [RESUME] bit.

2.7.8 Handling Private Transmit (Master Read) Transfers

The data transmission for a private master read is initiated by issuing a TX transfer command. The TX transfer command can be issued by writing the TX command data structure in to the COMMAND_QUEUE_PORT register.

The slave responds for a private read address with ACK when all of the following conditions are met:

- TX Command is valid in the Command Queue
- TX FIFO has data equal to either the data length size of the command or the TX_START_THLD size is met.
- Response Queue Non-Full

Otherwise, the slave responds with NACK for the private read address.

To determine for which of the condition the NACK occurred, the controller provides further information as follows:

- An additional interrupt INTR_STATUS[READ_REQ_RECV_STS] is asserted when there is no valid command in the Command Queue.
- The CCC_DEVICE_STATUS[DATA_NOT_READY] bit is set when the data in the TX FIFO is not equal to data length size of the command or the TX_START_THLD.
- The CCC_DEVICE_STATUS[DATA_NOT_READY] bit is also set if Response Queue is full.

During HDR transfers (ENTHDR CCC detected), the slave responds with either "Slave NACK" preamble (in DDR mode) or with the exit/Restart pattern (in TSL/TSP mode) to indicate the Slave's inability to transmit the data for the private read command when one of the conditions are not met.

Once the read address is acknowledged with ACK (accepted), the controller expects the application to provide enough data in the TX FIFO as specified in the TX command to avoid underrun condition.

In the PIO mode (non DMA) of operation, use the TX_BUF_EMPTY_THLD interrupt to fill the TX FIFO while the TX data is being transmitted on the I3C bus.

In the DMA mode of operation, the DMA request of the handshake interface is initiated when the TX FIFO empty location level reaches the programmed TX_BUF_EMPTY_THLD level.

When an underrun condition is encountered (when the system latency is high to provide the transmit data), the controller sets the UNDERFLOW_ERR bit of the CCC_DEVICE STATUS register and terminates the transfer on the I3C bus.

The controller, once it sets the UNDERFLOW_ERR bit, rejects (NACK) any further private transfer request from Master until the Master reads the device status through a GETSTATUS CCC and the slave application resumes the slave operation by setting the bit DEVICE_CTRL [RESUME] bit of the register.

If the transmit FIFO does not have threshold amount of data to respond for the master read request, the controller NACKs the request and in CCC_DEVICE_STATUS[DATA_NOT_READY] bit is set. The controller, once it sets the DATA_NOT_READY bit, rejects (NACK) any further private read transfer request from Master until data is available in the transmit FIFO.



Note In I2C mode of operation, if an underflow error is encountered in a read transfer, controller cannot terminate the ongoing transfer. The controller rejects (NACK) any further private transfer request from Master until the slave application resumes the slave operation by setting the bit DEVICE_CTRL [RESUME].

2.7.9 Hot-Join Request Generation

2.7.9.1 Overview of Hot-Join Request Generation with DWC_mipi_i3c Slave

The Slave controller always attempts to issue the first Hot-Join request to the current master when the following conditions are met:

1. Configured as a Hot-Join capable device
2. SLV_EVENT_STATUS[HJ_EN] is 1.
3. Dynamic address is invalid (not assigned yet)
4. Hot-Join event generation is not disabled by the Master (through DISEC CCC transfer)
5. Bus Idle condition is met (1ms) as programmed in the Bus Timing Register.

If a NACK response is received or when an arbitration loss is encountered after the first attempt, the Slave controller attempts to re-send the Hot-join request in the next START condition seen on the bus, or on meeting Bus Idle condition (whichever happens first), provided conditions 3 and 4 are still valid.

In the Adaptive mode (mode_i2c = 0), the Slave controller does not generate the Hot-Join request until I3C Broadcast address (7h7E) is seen on the I3C bus, if the DEVICE_CTRL[ADAPTIVE_I2C_I3C] is set. If DEVICE_CTRL[ADAPTIVE_I2C_I3C] is set to 0, the Slave controller generates the Hot-Join when the previously specified conditions to generate Hot-join are met. It is better to set ADAPTIVE_I2C_I3C to 1, if the slave application does not have a prior knowledge of the bus (I2C/I3C) to which the controller is connected. If DEVICE_CTRL[ADAPTIVE_I2C_I3C] is set to 1, the Slave controller does not send the Hot-Join until it confirms the bus mode as I3C indicated by receipt of 7E header on the line.

Application can program SLV_EVENT_STATUS[HJ_EN] to 0 to disable the HJ capability of the controller. When this bit is set to 0, controller does not generate HJ request and behaves as if its a non-HJ device. When this bit is set to 1, controller generates HJ request when all the conditions mentioned are met. Note that programming of SLV_EVENT_STATUS[HJ_EN] is a one time task, which is done only during the initial configuration and when the controller is disabled.



Note When Hot Join is enabled ($\text{SLV_EVENT_STATUS}[\text{HJ_EN}] = 1$), controller does not participate in ENTDAA until HJ request is sent on the Bus.

2.7.9.2 Enabling Hot-Join Request Generation with DWC_mipi_i3c Slave

To enable Hot-Join request, enable the parameter *Hot Join Support* under the Slave Configuration tab, during the *Specify Configuration* Activity in coreConsultant.

2.7.10 Slave Interrupt Request Generation

2.7.10.1 Overview of Slave Interrupt Request (SIR) Generation

The DWC_mipi_i3c controller can generate the Slave Interrupt Request (SIR), which is an In-Band Interrupt (IBI), to get the attention of the Current Master.

The DWC_mipi_i3c controller allows the slave application to generate the SIR through a particular register.

2.7.10.2 Description of Slave Interrupt Request (SIR) Generation

The application of the DWC_mipi_i3c controller can decide to send a Slave Interrupt Request by asserting the SIR bit in the Slave Interrupt Request Register, along with the type of IBI to be generated, which can be selected by SIR_CTRL bits. If the SIR_CTRL bits are set to 2'b00, the IBI is generated after the BUS FREE TIME programmed in the Bus Timing register has expired, or on the next available START condition on the I3C bus.



Note Only SIR_CTRL- 2'b00 is supported. Other values are reserved.

The status of the IBI generation is updated in the IBI_STS field of Slave Interrupt Request register (SLV_INTR_REQ), which is informed to the application by the IBI_UPDATED_STS interrupt in INTR_STATUS register by DWC_mipi_i3c controller. On successful completion of IBI_STS update, the SIR bit in SLV_INTR_REQ register is auto cleared. On receiving this interrupt, the Slave application can read the IBI_STS field in the Slave Interrupt Request register.

Table 2-47 Values of IBI_STS

IBI_STS	Value	Description
Reserved	00	Default Value
Success	01	SIR accepted by the Master (ACK response received)
Reserved	10	Reserved
Not Attempted	11	SIR not attempted

The controller does not attempt to issue the IBI and generates the 'Not Attempted (2'b11)' status under following conditions:

- Master has not assigned the Dynamic Address.
- Master has cleared the assigned Dynamic Address through RSTDAA.
- Master has disabled the SIR_EN through DISEC CCC (SIR_EN in SLV_EVENT_STATUS register).
- The controller has switched the role to Master (applicable only for secondary master configuration).

2.7.10.3 Slave Interrupt Request (SIR) Related Registers

The following are the SIR-related registers in DWC_mipi_i3c (these registers are also explained in [Chapter 5, "Register Descriptions"](#)):

- IBI_UPDATED_STAT
- SLV_INTR_REQ
- SLV_EVENT_STATUS
- BUS_IDLE_TIMING

2.7.11 Master Request Generation

2.7.11.1 Overview of Master Request (MR) Generation

The DWC_mipi_i3c controller can generate the Master Request (MR) in non-current master mode to request for I3C bus ownership from the Current Master.

The DWC_mipi_i3c controller allows the slave application to generate the MR through a particular register.

2.7.11.2 Description of Master Request (MR) Generation

The application of the DWC_mipi_i3c controller can decide to send a Master Request by asserting the MR bit in the Slave Interrupt Request Register.

The status of the IBI generation is updated in the IBI_STS field of Slave Interrupt Request Register (SLV_INTR_REQ), which is informed to the application by the IBI_UPDATED_STS interrupt in INTR_STATUS register by DWC_mipi_i3c controller. Also after IBI_STS is updated the MR bit in SLV_INTR_REQ register is auto cleared. On receiving this interrupt, the Slave application can read the IBI_STS field in the Slave Interrupt Request register.

Table 2-48 Values of IBI_STS

IBI_STS	Value	Description
Reserved	00	Default Value
Success	01	MR accepted by the Master (ACK response received).
Reserved	10	Reserved
Not Attempted	11	MR not attempted

The controller does not attempt to issue the IBI and generates the 'Not Attempted (2'b11)' status under the following conditions.

- Master has not assigned the Dynamic Address.
- Master has cleared the assigned Dynamic Address through RSTDAA.
- Master has disabled the MR_EN through DISEC CCC (MR_EN in SLV_EVENT_STATUS register).
- The controller has switched the role to Master (applicable only for secondary master configuration).

2.7.11.3 Master Request (MR) Related Registers

The following are the MR related registers in DWC_mipi_i3c (these registers are also explained in [Chapter 5, "Register Descriptions"](#)):

- IBI_UPDATED_STAT
- SLV_INTR_REQ
- SLV_EVENT_STATUS
- BUS_FREE_AVAIL_TIMING



Note The application must not assert both Slave Interrupt Request and Master Request together. Once SIR/MR field in SLV_INTR_REQ register is asserted, the application should wait until IBI status is updated by the controller through IBI_UPDATED_STAT interrupt (also SIR/MR bit gets auto cleared), before asserting SIR/MR again.

2.7.12 Disabling DWC_mipi_i3c Slave

The application can disable the DWC_mipi_i3c when operating as a Slave at any time by clearing the DEVICE_CTRL[ENABLE] bit. The application should then poll the DEVICE_CTRL[ENABLE] bit until it turns to 1'b0 to confirm that the controller is in disabled state. If the Slave controller is busy in executing any I3C bus transfers (like receiving/transmitting an I3C transfer or transmitting an IBI), then the controller enters the disabled state only after the controller completes the present on-going transfers. Once the controller enters the disabled state, it responds with NACK for any address match including '7h7E.



Note Slave IP should not be Disabled when Status for SIR/MR request is pending.

2.7.13 Data Structure in DWC_mipi_i3c Slave

2.7.13.1 Transmit Command Data Structure

The Transmit Command Structure in DWC_mipi_i3c Slave is used to respond with data for a private read command from the Current Master.

Table 2-49 Transmit Command Data Structure

Bits	Name	Memory Access	Description
31:16	DATA_LENGTH	W	Data Length This field is used to indicate the data length of the master read transfer.
15:6	RSVD	W	Reserved for Future use
5:3	TID	W	Transmit Transaction ID This field is used as the identification tag for the command.
2:0	CMD_ATTR	W	Command Attribute Defines the Command attribute and its field format. <ul style="list-style-type: none"> ■ 0 - Transmit Command without IBI ■ 1-7 - Reserved

2.7.13.2 Response Data Structure

The Response Structure DWC_mipi_i3c APB Slave is used to indicate the completion of the transmit command (Private Master Read operation) or the completion of Master Write operation (Private Master Write operation).

Table 2-50 Response Data Structure

Bits	Name	Memory Access	Description
31:28	ERR_STATUS	R	<ul style="list-style-type: none"> ■ 0: No Error ■ 1: CRC error (master write in DDR mode) ■ 2: Parity error (master write in both DDR and SDR mode) ■ 3: Frame error (HDR mode master write) ■ 6: Underflow/Overflow error ■ 10: Master early termination
27	RX_RSP	R	<p>Transaction Type This field is used to identify the type of transaction:</p> <ul style="list-style-type: none"> ■ 0: Transmit Response ■ 1: Receive Response
26:24	TID	R	Transmit Transaction ID This field is used as the identification tag for the transmit command.

Table 2-50 Response Data Structure (Continued)

Bits	Name	Memory Access	Description
23:16	CCC_HDR_HEADER	R	<p>HDR Command Code</p> <ul style="list-style-type: none"> ■ 8'h00 - SDR Transfer ■ Others - HDR Command Code or in the case of Secondary master, whenever DEFSLV CCC is received, the DEFSLV CCC command code is reflected in this field. The application comes to know that this field reflects the CCC command code if TID field is set to 3'b111 along with RX_RESP field set to 1'b1.
15:0	DATA_LENGTH	R	Data Length in bytes of Master Write Transfer or the Remaining Length of Master Read Transfer (Master termination or Slave underrun). In case of DEFSLVS which is indicated by TID and RX_RESP field all set to one, this field indicates the device count to the application.



Slave Data Length: The Data Length field in Response of a private read/write transfer that has experienced any error (has the appropriate Error Status flag set in Response) does not accurately reflect the number of bytes sent or received before the occurrence of the error.

Since the Error Handling flow recommends that the appropriate queues be Flushed the Data length Reported has no particular significance.

3

Parameter Descriptions

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.

- Role Configuration on [page 154](#)
- HCI Configuration on [page 155](#)
- Basic Configuration on [page 157](#)
- Queues and Interfaces on [page 158](#)
- Master Configuration on [page 159](#)
- Slave Configuration on [page 161](#)
- Clock(s) Configuration on [page 165](#)
- External Memory on [page 172](#)
- Preset Values on [page 173](#)
- HCI Preset Values on [page 185](#)

3.1 Role Configuration Parameters

Table 3-1 Role Configuration Parameters

Label	Description
Configure DWC_mipi_i3c	
Device Role	<p>Specifies the Device Role of DWC_mipi_i3c as</p> <ul style="list-style-type: none"> ■ Master-Only ■ Slave-Only ■ Programmable Master and Slave ■ Secondary Master ■ Slave-Lite <p>NOTE: Programmable Master-Slave configuration will not be supported in upcoming releases. Please select Secondary Master configuration if you want the DWC_mipi_i3c controller to operate in both Master and Slave roles.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Master-Only (1) ■ Slave-Only (4) ■ Programmable Master-Slave (2) ■ Secondary Master (3) ■ Slave-Lite (5) <p>Default Value: (([<functionof>]) == 1) ? 3:5</p> <p>Enabled: ([<functionof>]) == 1</p> <p>Parameter Name: IC_DEVICE_ROLE</p>

3.2 HCI Configuration Parameters

Table 3-2 HCI Configuration Parameters

Label	Description
HCI compliance	
Host Controller Interface(HCI)	<p>Specifies DWC_mipi_i3c supports HCI compliant or not 0: HCI Not compliant or SNPS Properity RegisterMap 1: HCI Compliant</p> <p>Values:</p> <ul style="list-style-type: none"> ■ false (0) ■ true (1) <p>Default Value: false</p> <p>Enabled: IC_DEVICE_ROLE==1</p> <p>Parameter Name: IC_HAS_HCI</p>
Host Controller Interface Section offset Settings	
Operational Register Section Offset	<p>Default value of Operational Register Section Offset.</p> <p>Values: 0x0, ..., 0xffff</p> <p>Default Value: 0x0</p> <p>Enabled: Always</p> <p>Parameter Name: IC_DFLT_OPERATION_REG_OFFSET</p>
PIO Section Offset	<p>Default value of PIO Section Offset.</p> <p>Values: 0x0, ..., 0xffff</p> <p>Default Value: 0x300</p> <p>Enabled: Always</p> <p>Parameter Name: IC_DFLT_PIO_SEC_OFFSET</p>
Ring Headers Section Offset	<p>Default value of Ring Headers Section Offset.</p> <p>Values: 0x0, ..., 0xffff</p> <p>Default Value: 0x200</p> <p>Enabled: Always</p> <p>Parameter Name: IC_DFLT_RING_HDR_SEC_OFFSET</p>
Extended Capabilities Section Offset	<p>Default value of Extended Capabilities Section Offset.</p> <p>Values: 0x0, ..., 0xffff</p> <p>Default Value: 0x100</p> <p>Enabled: Always</p> <p>Parameter Name: IC_DFLT_EXTCAPS_SEC_OFFSET</p>
DAT Section Offset	<p>Default value of DAT Section Offset.</p> <p>Values: 0x0, ..., 0xffff</p> <p>Default Value: (IC_HAS_DAT==1) ? 0x400 : 0x000</p> <p>Enabled: IC_HAS_DAT==1</p> <p>Parameter Name: IC_DFLT_DAT_SEC_OFFSET</p>

Table 3-2 HCI Configuration Parameters (Continued)

Label	Description
DCT Section Offset	<p>Default value of DCT Section Offset.</p> <p>Values: 0x0, ..., 0xffff</p> <p>Default Value: (IC_HAS_DAT==1) ? 0x600 : 0x000</p> <p>Enabled: IC_HAS_DAT==1</p> <p>Parameter Name: IC_DFLT_DCT_SEC_OFFSET</p>

3.3 Basic Configuration Parameters

Table 3-3 Basic Configuration Parameters

Label	Description
HDR Support	
HDR-DDR	<p>Configures DWC_mipi_i3c to have high data rate - Double Data Rate Mode support. If this parameter is selected, DWC_mipi_i3c transfers the data on both edge transitions of the SCL for generation of high data rate.</p> <p>Values: 0, 1</p> <p>Default Value: 0</p> <p>Enabled: Always</p> <p>Parameter Name: IC_SPEED_HDR_DDR</p>
HDR-TS	
	<p>Configures DWC_mipi_i3c to have high data rate - Ternary Symbol Mode support. If this Parameter is selected, DWC_mipi_i3c transfers the data as Ternary symbols by using both SCL and SDA lines as data lines. DWC_mipi_i3c requires the recovered clock as an external input to sample the data. You can use either CDR block or Pure Digital DPLL solutions to recover the clock. Note that these clock recover mechanisms are not a part of DWC_mipi_i3c deliverables and are expected to be implemented outside the IP boundary.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ false (0) ■ true (1) <p>Default Value: false</p> <p>Enabled: IC_DEVICE_ROLE<5</p> <p>Parameter Name: IC_SPEED_HDR_TS</p>
PEC Support	
PEC Support Enable	<p>Configures DWC_mipi_i3c to support Packet Error Check (PEC) in SDR transfers. If this parameter is selected, DWC_mipi_i3c (In Master-only and Slave-lite configurations) will have capability to generate and validate PEC Byte in</p> <ul style="list-style-type: none"> ■ SDR Broadcast and Directed CCC Transfers, ■ SDR Private Write and Read Transfers and ■ SDR IBI Transfers <p>as per the JEDEC Sideband specification.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ false (0) ■ true (1) <p>Default Value: false</p> <p>Enabled: ((IC_DEVICE_ROLE==1) (IC_DEVICE_ROLE==5)) && (IC_HAS_HCI==0)</p> <p>Parameter Name: IC_HAS_PEC</p>

3.4 Queues and Interfaces Parameters

Table 3-4 Queues and Interfaces Parameters

Label	Description
Additional Interfaces	
DMA Handshaking Interface	<p>When set to true, DMA handshaking interface signals are provided for transmit and receive queues at the top-level I/O.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ false (0) ■ true (1) <p>Default Value: false</p> <p>Enabled: IC_DEVICE_ROLE<5</p> <p>Parameter Name: IC_HAS_DMA</p>
Buffer Sizing	
Buffer Level Selection	<p>Sets the depth of the Command, Response, Transmit and Receive Buffers</p> <p>MINIMUM - Commands - 2(4 locations), Response - 2(2 locations), Transmit - 16(16 locations), Receive - 16(16 locations)</p> <p>TYPICAL - Commands - 4(8 locations), Response - 4(4 locations), Transmit - 32(32 locations), Receive - 32(32 locations)</p> <p>MAXIMUM - Commands - 8(16 locations), Response - 8(8 locations), Transmit - 64(64 locations), Receive - 64(64 locations)</p> <p>NOTE: Commands and Response selection indicates the number of commands or response that can be pipelined. The transmit and receive buffer depth selections are based on number of location. Each location can hold 4 bytes of data.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Minimum (2 Commands,2 Response,16 Transmit,16 Receive) (1) ■ Typical (4 Commands,4 Response,32 Transmit,32 Receive) (2) ■ Maximum (8 Commands,8 Response,64 Transmit,64 Receive) (3) <p>Default Value: Typical (4 Commands,4 Response,32 Transmit,32 Receive)</p> <p>Enabled: IC_DEVICE_ROLE!=5</p> <p>Parameter Name: IC_BUF_LVL_SEL</p>

3.5 Master Configuration Parameters

Table 3-5 Master Configuration Parameters

Label	Description
Master Mode Configuration	
No Of Addressable Devices	<p>Selects the number of DWC_mipi_i3c devices present in the DWC_mipi_i3c system. This parameter is used to calculate the default value of the Device Characteristics Table depth ($IC_DEV_CHAR_TABLE_BUF_DEPTH = IC_NUM_DEVICES * 4$). During ENTDAA procedure, the controller receives information from the devices and stores it in the Device Characteristics Table. It takes four locations in the DCT to store information from one slave.</p> <p>Values: 1, ..., 11 Default Value: 2 Enabled: $IC_DEVICE_ROLE < 4$ Parameter Name: <code>IC_NUM_DEVICES</code></p>
Device Address Table Depth	<p>Sets the depth of the Device Address Table Buffer.</p> <p>Values: 1, ..., $((IC_HAS_HCI == 1) ? 22 : 11)$ Default Value: $(IC_HAS_HCI == 1) ? (IC_NUM_DEVICES * 2) : (IC_NUM_DEVICES)$ Enabled: $(IC_HAS_DAT == 1) \&& (IC_DEVICE_ROLE < 4)$ Parameter Name: <code>IC_DEV_ADDR_TABLE_BUF_DEPTH</code></p>
Device Characteristics Table Depth	<p>Sets the depth of the Device Characteristics Table Buffer.</p> <p>Values: 1, ..., 44 Default Value: $(IC_DEVICE_ROLE < 4) ? (IC_NUM_DEVICES * 4) : IC_NUM_DEVICES$ Enabled: $(IC_HAS_DAT == 1) \&& (IC_DEVICE_ROLE < 4)$ Parameter Name: <code>IC_DEV_CHAR_TABLE_BUF_DEPTH</code></p>
Master IBI settings	
Support IBI with data	<p>Configures DWC_mipi_i3c master to have IBI with data support. When this parameter is enabled, the controller generates clock for receiving data from the slave following the SIR Interrupt. IBI with data feature will be enabled by default (cannot be deselected) in future releases.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ false (0) ■ true (1) <p>Default Value: $(IC_HAS_HCI == 1) ? 1 : 0$ Enabled: $(IC_DEVICE_ROLE == 1) \&& (IC_OPEN_DRAIN_CLASS_PULLUP == 1)$ Parameter Name: <code>IC_HAS_IBI_DATA</code></p>

Table 3-5 Master Configuration Parameters (Continued)

Label	Description
IBI Buffer Level Selection	<p>Sets the depth of the IBI status and data buffers. IBI data buffer depth is applicable only in Master-Only config (IC_DEVICE_ROLE=1)</p> <p>MINIMUM - IBI Status - 4(4 locations), IBI Data Payload - 16(16 locations)</p> <p>TYPICAL - IBI Status - 8(8 locations), IBI Data Payload - 32(32 locations)</p> <p>MAXIMUM - IBI Status - 16(16 locations), IBI Data Payload - 64(64 locations)</p> <p>NOTE: Status depth selection indicates the number of IBI status that can be stored in IBI status buffer. The IBI data depth selection is based on number of location. Each location can hold 4 bytes of data.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Minimum (4 IBI Status, 16 IBI Payload) (1) ■ Typical (8 IBI Status, 32 IBI Payload) (2) ■ Maximum (16 IBI Status, 64 IBI Payload) (3) <p>Default Value: Minimum (4 IBI Status, 16 IBI Payload)</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_IBI_BUF_LVL_SEL</p>
IBHR Patterns Settings	
Device Reset Support	<p>Configures DWC_mipi_i3c to have Device Reset Support. When this parameter is enabled, the controller generates the following Reset Patterns.</p> <ul style="list-style-type: none"> ■ HDR Exit-Pattern, ■ SCL Low Pattern. <p>Values:</p> <ul style="list-style-type: none"> ■ false (0) ■ true (1) <p>Default Value: ((IC_HAS_HCI==1) && (IC_HAS_DAT==0)) ? 1 :0</p> <p>Enabled: (IC_HAS_HCI==0) (IC_HAS_DAT==0)</p> <p>Parameter Name: IC_HAS_DEVICE_RESET_SUPPORT</p>

3.6 Slave Configuration Parameters

Table 3-6 Slave Configuration Parameters

Label	Description
Bus Characteristic Register Configuration	
Bridge Device	<p>Specifies whether DWC_mipi_i3c is a bridge device or not.</p> <ul style="list-style-type: none"> ■ 0: Not a bridge device ■ 1: Bridge device <p>Note: Bridge device is not supported.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ false (0) ■ true (1) <p>Default Value: false</p> <p>Enabled: 0</p> <p>Parameter Name: IC_SLV_BRIDGE</p>
Offline Capable	<p>Specifies whether DWC_mipi_i3c is offline capable or not.</p> <ul style="list-style-type: none"> ■ 0: Not offline capable ■ 1: Offline capable <p>Note: Offline capability is not supported</p> <p>Values:</p> <ul style="list-style-type: none"> ■ false (0) ■ true (1) <p>Default Value: false</p> <p>Enabled: 0</p> <p>Parameter Name: IC_SLV_OFFLINE_CAP</p>
IBI Support	<p>Specifies whether DWC_mipi_i3c supports Slave In-Band Interrupt or not.</p> <ul style="list-style-type: none"> ■ 0: Slave In-Band Interrupt is not supported ■ 1: Slave In-Band Interrupt is supported <p>If this parameter is selected the DWC_mipi_i3c Slave will support both SIR and Hot-Join. The configuration parameter for Hot-join, IC_SLV_HJ, will be automatically set to 1, if IC_SLV_IBI is set.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ false (0) ■ true (1) <p>Default Value: false</p> <p>Enabled: IC_DEVICE_ROLE>1</p> <p>Parameter Name: IC_SLV_IBI</p>

Table 3-6 Slave Configuration Parameters (Continued)

Label	Description
Max Data Speed Limitation	<p>Specifies whether or not DWC_mipi_i3c has maximum data speed limitation. Note that this bit can be programmed/overwritten by software. If it is unclear whether or not your design has Maximum Data Speed Limitation, select "Include programming for maxRd, maxWr and tsco" to have the flexibility of providing these values through programmable registers.</p> <ul style="list-style-type: none"> ■ 0: No limitation ■ 1: Limitation <p>Values:</p> <ul style="list-style-type: none"> ■ false (0) ■ true (1) <p>Default Value: false</p> <p>Enabled: IC_DEVICE_ROLE>1</p> <p>Parameter Name: IC_SLV_DATA_SPEED_LIMIT</p>
Max Data Speed Configuration	
Programmable maxRd maxWr tSCO	<p>When Enabled input ports and register programming options are made available for maxRd (maximum read data rate) maxWr(maximum write data rate) and tsco (clock to data turnaround time). These values are returned by the slave when it received GETMXDS CCC. This will require that you will have to know post silicon characterized values of maxRd, maxWr and tsco. It is strongly recommended to select this feature to have the flexibility of providing these values via programmable registers after post silicon characterization.</p> <p>Values: 0, 1</p> <p>Default Value: ((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)) ? 1 : 0</p> <p>Enabled: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)</p> <p>Parameter Name: IC_SLV_MXDS_PROG</p>
Maximum Sustained Write Data Rate	<p>Specifies the Maximum Sustained Data Rate for non-CCC messages sent by Master Device to DWC_mipi_i3c Slave device.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ fSCL Max(12.5 MHz) (0) ■ 8MHz (1) ■ 6MHz (2) ■ 4MHz (3) ■ 2MHz (4) ■ Reserved_5 (5) ■ Reserved_6 (6) ■ Reserved_7 (7) <p>Default Value: fSCL Max(12.5 MHz)</p> <p>Enabled: IC_SLV_MXDS_PROG==0</p> <p>Parameter Name: IC_SLV_MXDS_MAX_WR_SPEED</p>

Table 3-6 Slave Configuration Parameters (Continued)

Label	Description
Maximum Sustained Read Data Rate	<p>Specifies the Maximum Sustained Data Rate for non-CCC messages sent by DWC_mipi_i3c Slave Device to Master device.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ fSCL Max(12.5 MHz) (0) ■ 8MHz (1) ■ 6MHz (2) ■ 4MHz (3) ■ 2MHz (4) ■ Reserved_5 (5) ■ Reserved_6 (6) ■ Reserved_7 (7) <p>Default Value: fSCL Max(12.5 MHz)</p> <p>Enabled: IC_SLV_MXDS_PROG==0</p> <p>Parameter Name: IC_SLV_MXDS_MAX_RD_SPEED</p>
Clock to Data Turnaround Time	<p>Specifies the clock to data turnaround time (Tsc0 parameter) of DWC_mipi_i3c Slave device.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 8ns (0) ■ 9ns (1) ■ 10ns (2) ■ 11ns (3) ■ 12ns (4) ■ Reserved_5 (5) ■ Reserved_6 (6) ■ Reserved_7 (7) <p>Default Value: 8ns</p> <p>Enabled: IC_SLV_MXDS_PROG==0</p> <p>Parameter Name: IC_SLV_MXDS_CLK_DATA_TURN</p>
Maximum Read Turnaround Time	<p>Specifies the maximum read turnaround time (in microseconds (us)) of DWC_mipi_i3c Slave Lite.</p> <p>Values: 0x0, ..., 0xfffff</p> <p>Default Value: 0x0</p> <p>Enabled: Always</p> <p>Parameter Name: IC_SLV_MXDS_MAX_RD_TURN</p>

Table 3-6 Slave Configuration Parameters (Continued)

Label	Description
Default Values	
MWL Value	<p>Default Maximum Write Length value of DWC_mipi_i3c. Sets the reset value of Maximum Write Length register of DWC_mipi_i3c. This value is returned by the DWC_mipi_i3c Slave if the Master sends a GETMWL CCC. The Maximum Write Length register is overwritten by a new MWL value if the Master sends SETMWL CCC.</p> <p>Values: 0x0, ..., 0xffff Default Value: 0xff Enabled: IC_DEVICE_ROLE>1 Parameter Name: IC_SLV_DFLT_MWL</p>
MRL Value	<p>Default Maximum Read Length value of DWC_mipi_i3c. Sets the reset value of Maximum Read Length register of DWC_mipi_i3c. This value is returned by the DWC_mipi_i3c Slave if the Master sends a GETMRL CCC. The Maximum Read Length register is overwritten by a new MRL value if the Master sends SETMRL CCC.</p> <p>Values: 0x0, ..., 0xffff Default Value: 0xff Enabled: IC_DEVICE_ROLE>1 Parameter Name: IC_SLV_DFLT_MRL</p>

3.7 Clock(s) Configuration Parameters

Table 3-7 Clock(s) Configuration Parameters

Label	Description
Clock Relationship Configuration	
Application Clock to Core Clock	<p>Specifies the relationship between Application Slave interface clock and core_clk.</p> <ul style="list-style-type: none"> ■ Identical (0): Clocks are identical(same source); no metastability flops are used for data passing between clock domains. ■ Asynchronous (1): Clocks can be completely asynchronous to each other; metastability flops are used for data passing between clock domains. <p>Values:</p> <ul style="list-style-type: none"> ■ Identical (0) ■ Asynchronous (1) <p>Default Value: Identical</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_CLK_TYPE</p>
DMA Clock to Core Clock	<p>Specifies the relationship between DMA clock and core_clk.</p> <ul style="list-style-type: none"> ■ Identical (0): Clocks are identical(same source); no metastability flops are used for data passing between clock domains. ■ Asynchronous (1): Clocks can be completely asynchronous to each other; metastability flops are used for data passing between clock domains. <p>Values:</p> <ul style="list-style-type: none"> ■ Identical (0) ■ Asynchronous (1) <p>Default Value: Identical</p> <p>Enabled: IC_HAS_DMA==1 && IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DMA_CORE_CLK_TYPE</p>

Table 3-7 Clock(s) Configuration Parameters (Continued)

Label	Description
DMA Clock to Application Clock	<p>Specifies the relationship between DMA clock and Application Slave Interface Clock.</p> <ul style="list-style-type: none"> ■ Identical (0): Clocks are identical(same source); no metastability flops are used for data passing between clock domains. ■ Asynchronous (1): Clocks can be completely asynchronous to each other; metastability flops are used for data passing between clock domains. <p>Values:</p> <ul style="list-style-type: none"> ■ Identical (0) ■ Asynchronous (1) <p>Default Value: ((IC_CLK_TYPE==0) && (IC_DMA_CORE_CLK_TYPE==0)) ? 0:1</p> <p>Enabled: (IC_HAS_DMA==1) && ((IC_DEVICE_ROLE<3) (IC_DEVICE_ROLE==4)) && (IC_CLK_TYPE==1) && (IC_DMA_CORE_CLK_TYPE==1)</p> <p>Parameter Name: IC_DMA_SLAVE_CLK_TYPE</p>
HDR Clock to Core Clock	<p>Specifies the relationship between HDR clock and core_clk.</p> <ul style="list-style-type: none"> ■ Identical (0): Clocks are identical(same source); no metastability flops are used for data passing between clock domains. ■ Asynchronous (1): Clocks can be completely asynchronous to each other; metastability flops are used for data passing between clock domains. <p>Values:</p> <ul style="list-style-type: none"> ■ Identical (0) ■ Asynchronous (1) <p>Default Value: Asynchronous</p> <p>Enabled: 0</p> <p>Parameter Name: IC_HDR_CLK_TYPE</p>
Synchronizer(s) depth settings	
Synchronizer depth settings	<p>Defines the number of synchronization register stages for signals passing from one clock domain to another asynchronous clock domain. All the asynchronous clock domain crossings will have same number of synchronization register stages.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge ■ (3) Three-stage synchronization: All stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: Two-stage (Both on positive edge)</p> <p>Enabled: (IC_CLK_TYPE==1) ((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)) (IC_SPEED_HDR_TS==1)</p> <p>Parameter Name: IC_SYNC_DEPTH</p>

Table 3-7 Clock(s) Configuration Parameters (Continued)

Label	Description
Application Clock to Core Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c Application slave clock domain to DWC_mipi_i3c Core Clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge ■ (3) Three-stage synchronization: All stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_SLVIF_2_COREIF_SYNC_DEPTH</p>
Core Clock to Application Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c Core Clock domain to DWC_mipi_i3c Application Slave clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge ■ (3) Three-stage synchronization: All stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_COREIF_2_SLVIF_SYNC_DEPTH</p>
Core Clock to DMA Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c Core Clock domain to DWC_mipi_i3c DMA clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge ■ (3) Three-stage synchronization: All stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_COREIF_2_DMAIF_SYNC_DEPTH</p>

Table 3-7 Clock(s) Configuration Parameters (Continued)

Label	Description
DMA Clock to Core Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c DMA Clock domain to DWC_mipi_i3c Core clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge ■ (3) Three-stage synchronization: All stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_DMAIF_2_COREIF_SYNC_DEPTH</p>
Application Clock to DMA Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c Slave Clock domain to DWC_mipi_i3c DMA clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge ■ (3) Three-stage synchronization: All stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_SLVIF_2_DMAIF_SYNC_DEPTH</p>
DMA Clock to Application Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c DMA Clock domain to DWC_mipi_i3c Slave clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge ■ (3) Three-stage synchronization: All stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_DMAIF_2_SLVIF_SYNC_DEPTH</p>

Table 3-7 Clock(s) Configuration Parameters (Continued)

Label	Description
HDR clock to Core clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c HDR clock domain to DWC_mipi_i3c Core Clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge ■ (3) Three-stage synchronization: All stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_HDRIF_2_COREIF_SYNC_DEPTH</p>
SCL Clock to Core Clock	<p>Defines the number of synchronization register stages for signals passing from the SCL Clock domain to DWC_mipi_i3c core clock or application clock (depending on configuration) domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_SCLIF_2_COREIF_SYNC_DEPTH</p>
Core Clock to SCL Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c core clock domain or application clock (depending on configuration) to DWC_mipi_i3c SCL clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_COREIF_2_SCLIF_SYNC_DEPTH</p>

Table 3-7 Clock(s) Configuration Parameters (Continued)

Label	Description
SCL Clock to HDR Tx Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c SCL Clock domain to DWC_mipi_i3c HDR Tx clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_SCLIF_2_HDRTXIF_SYNC_DEPTH</p>
SCL Clock to HDR Rx Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c SCL Clock domain to DWC_mipi_i3c HDR Rx clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_SCLIF_2_HDRRXIF_SYNC_DEPTH</p>
SDA Clock to HDR Tx Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c SDA Clock domain to DWC_mipi_i3c HDR Tx clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_SDAIF_2_HDRTXIF_SYNC_DEPTH</p>
SDA Clock to HDR Rx Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c SDA Clock domain to DWC_mipi_i3c HDR Rx clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_SDAIF_2_HDRRXIF_SYNC_DEPTH</p>

Table 3-7 Clock(s) Configuration Parameters (Continued)

Label	Description
HDR Rx Clock to SCL Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c HDR Rx Clock domain to DWC_mipi_i3c SCL clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_HDERRXIF_2_SCLIF_SYNC_DEPTH</p>
HDR Rx to HDR Tx Clock	<p>Defines the number of synchronization register stages for signals passing from the DWC_mipi_i3c HDR Rx Clock domain to DWC_mipi_i3c HDR Tx clock domain.</p> <ul style="list-style-type: none"> ■ (2) Two-stage synchronization: Both stages positive edge <p>Values:</p> <ul style="list-style-type: none"> ■ Two-stage (Both on positive edge) (2) ■ Three-stage (All on positive edge) (3) <p>Default Value: IC_SYNC_DEPTH</p> <p>Enabled: 0</p> <p>Parameter Name: IC_HDERRXIF_2_HDRTXIF_SYNC_DEPTH</p>

3.8 External Memory Parameters

Table 3-8 External Memory Parameters

Label	Description
External Memory Configuration	
External Memory Data Width	<p>Sets the data width of the External Memory Interface.</p> <p>Values: 32</p> <p>Default Value: 32</p> <p>Enabled: 0</p> <p>Parameter Name: IC_RAM_DATA_WIDTH</p>
External Memory Depth	<p>Sets the Depth of the External Memory Interface.</p> <p>Values: ((IC_DEVICE_ROLE<4)? 11:6), ..., ((IC_DEVICE_ROLE<4)? 310:2056)</p> <p>Default Value: ((IC_DEVICE_ROLE<4) ? ((IC_HAS_IBI_DATA==1) ? ((IC_HAS_DAT==1) ? (IC_CMD_BUF_DEPTH + IC_TX_BUF_DEPTH + IC_RX_BUF_DEPTH + IC_DEV_CHAR_TABLE_BUF_DEPTH + IC_DEV_ADDR_TABLE_BUF_DEPTH + IC_IBI_BUF_DEPTH + IC_IBI_DATA_BUF_DEPTH) : (IC_CMD_BUF_DEPTH + IC_TX_BUF_DEPTH + IC_RX_BUF_DEPTH + IC_IBI_BUF_DEPTH + IC_IBI_DATA_BUF_DEPTH)) : (IC_CMD_BUF_DEPTH + IC_TX_BUF_DEPTH + IC_RX_BUF_DEPTH + IC_DEV_CHAR_TABLE_BUF_DEPTH + IC_DEV_ADDR_TABLE_BUF_DEPTH)) : (IC_CMD_BUF_DEPTH + IC_TX_BUF_DEPTH + IC_RX_BUF_DEPTH))</p> <p>Enabled: 0</p> <p>Parameter Name: IC_RAM_DEPTH</p>
External Memory Address Width	<p>Sets the Address width of the External Memory Interface.</p> <p>Values: 3, ..., 12</p> <p>Default Value: [<functionof> IC_RAM_DEPTH]</p> <p>Enabled: 0</p> <p>Parameter Name: IC_RAM_ADDR_WIDTH</p>
Register Retiming on forward and return path to External Memory Interface	<p>Enables Register Re-timing on forward and return path to External Memory (SPRAM) Interface.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ false (0) ■ true (1) <p>Default Value: false</p> <p>Enabled: IC_DEVICE_ROLE!=5</p> <p>Parameter Name: IC_FW_RAM_RETIMING</p>

3.9 Preset Values Parameters

Table 3-9 Preset Values Parameters

Label	Description
Clock Period Configuration	
Core Clock Period	<p>Specifies the minimum period of 'core_clk' (ns integers only), that is, the maximum frequency of core clock that is intended to be fed to DWC_mipi_i3c controller. In Master, Programmable Master Slave and Secondary Master Configurations software can read this value from HW_CAPABILITY.CLOCK_PERIOD to program Timing Registers that are employed to meet various DWC_mipi_i3c Timing requirement.</p> <p>Values: 1, ..., (IC_DEVICE_ROLE<4 ? 8:40) Default Value: IC_DEVICE_ROLE<4 ? 8:10 Enabled: IC_DEVICE_ROLE<5 Parameter Name: IC_CLK_PERIOD</p>
HDR Tx Clock Period	<p>Specifies the minimum period of 'hdr_tx_clk' (ns integers only) that is intended to be fed to controller in TSP/TSL enabled Programmable Master-Slave, Secondary Master, and APB Slave Configurations. Software can read this value from HW_CAPABILITY.HDR_TX_CLOCK_PERIOD to program the SLV_TSX_SYMBL_TIMING register.</p> <p>Values: 1, ..., 40 Default Value: 40 Enabled: (IC_DEVICE_ROLE>1 && IC_DEVICE_ROLE<5) && (IC_SPEED_HDR_TS==1) Parameter Name: IC_HDR_TX_CLK_PERIOD</p>
DEVICE_CTRL	
Default value of Hot-Join Control Enable	<p>Default value of HOT_JOIN_CTRL field in DEVICE_CTRL register. This bit can be used as a global control in Master mode of operation to ACK/NACK the Hot-Join Request from the devices.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Disable (0) ■ Enable (1) <p>Default Value: Disable Enabled: IC_DEVICE_ROLE<4 Parameter Name: IC_DFLT_HJ_CTRL</p>

Table 3-9 Preset Values Parameters (Continued)

Label	Description
Default Value of I2C Slave Present	<p>Default value of I2C Slave Present control bit. This bit indicates whether the Legacy I2C devices are present in the bus. When set, DWC_mipi_i3c Master initiates TSL transfers in HDR-Ternary mode of operation.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Disable (0) ■ Enable (1) <p>Default Value: Disable Enabled: IC_DEVICE_ROLE<4 Parameter Name: IC_DFLT_I2C_SLAVE_PRESENT</p>
Default value of I3C Broadcast Address Inclusion	<p>Reset value of IBA_INCLUDE bit in DEVICE_CTRL register. I3C Broadcast Address Inclusion Enable bit is used to include the I3C Broadcast address for private Write/Read transfers to either I3C or Legacy I2C slaves.</p> <ul style="list-style-type: none"> ■ 0: I3C Broadcast Address is not included for the private transfers. ■ 1: I3C Broadcast Address is included for the private transfers. <p>Note: If I3C Broadcast Address is not included for the private transfers, then the In-band Interrupts driven from the slave may not get priority and in-turn they may get delayed.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Not Included (0) ■ Included (1) <p>Default Value: Not Included Enabled: IC_DEVICE_ROLE<4 Parameter Name: IC_DFLT_IBA_INC</p>
DEVICE_ADDR	
Default value of Device Dynamic Address Valid	<p>Reset value of DYNAMIC_ADDR_VALID bit in DEVICE_ADDR register. Dynamic Address Valid bit is used to control whether the DYNAMIC_ADDR is valid or not.</p> <ul style="list-style-type: none"> ■ In I3C Main Master Mode, set this bit to 1 as it self-assigns its dynamic address. ■ In Secondary Master and Slave Modes, the Controller sets this bit to 1 when Main Master assigns the Dynamic Address during ENTDA or SETDASA mechanism. <p>Values:</p> <ul style="list-style-type: none"> ■ Not valid (0x0) ■ Valid (0x1) <p>Default Value: IC_DEVICE_ROLE<3 ? 1:0 Enabled: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<4) Parameter Name: IC_DFLT_DYNAMIC_ADDR_VALID</p>

Table 3-9 Preset Values Parameters (Continued)

Label	Description
Default Value of Device Dynamic Address	<p>Reset value of Device Dynamic Address.</p> <p>Device Dynamic Address is used to program the Dynamic Address of device. The Controller uses this address to respond to the Master Transactions in DWC_mipi_i3c Interface Mode:</p> <ul style="list-style-type: none"> ■ In Main Master Mode, you should program its self assigned address. ■ In other modes of operation, the Main Master assigns this address during Dynamic Address Assignment mode (DAA) or during Hot-Join Mechanism. <p>Values: 0x0, ..., 0x7f Default Value: 0x0 Enabled: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<4) Parameter Name: IC_DFLT_DYNAMIC_ADDR</p>
QUEUE_THLD_CTRL	
Default In-Band Interrupt Status Threshold Value	<p>Reset value of IBI_STATUS_THLD field in QUEUE_THLD_CTRL register.</p> <p>Every In-Band Interrupt received(with or without MDB/payload) by DWC_mipi_i3c controller generates an IBI status. This field controls the number of IBI status entries (or greater) in the IBI queue that trigger the IBI_THLD_STS interrupt. Valid range is 0 to IC_IBI_BUF_DEPTH-1. If the programmed value is greater than the maximum suggested value, only the number of bits required to address the full buffer depth is considered. A value of 0 sets the threshold for 1 entry, and a value of N sets the threshold for N+1 entries.</p> <p>Values: 0x0, ..., IC_IBI_BUF_DEPTH Default Value: 0x1 Enabled: IC_DEVICE_ROLE<4 Parameter Name: IC_DFLT_IBI_STS_THLD</p>
Default In-Band Interrupt Buffer Threshold Value	<p>Reset value of IBI_DATA_THLD field in QUEUE_THLD_CTRL register.</p> <p>This field represents the IBI data segment size in Dwords (4 bytes). The minimum supported segment size is 1 (4 bytes) and the maximum supported size is IC_IBI_DATA_BUF_DEPTH-1. The IBI_DATA_THLD field enables the slicing of the incoming IBI data and generate individual status and thereby promotes the cut-through operation in reading out the IBI data.</p> <p>Values: 0, ..., IC_IBI_DATA_BUF_DEPTH Default Value: 0 Enabled: IC_HAS_IBI_DATA==1 Parameter Name: IC_DFLT_IBI_BUF_THLD</p>

Table 3-9 Preset Values Parameters (Continued)

Label	Description
Default Response Queue Threshold Value	<p>Reset value of RESP_BUF_THLD field in QUEUE_THLD_CTRL register. RESP_BUF_THLD controls the number of entries (or greater) in the Response Queue that trigger the RESP_READY_STAT_INTR interrupt. Valid range is 0 to IC_RESP_BUF_DEPTH-1. If programmed value is greater than the maximum suggested value, only the number of bits required to address the full buffer depth is considered. A value of 0 sets the threshold for 1 entry, and a value of N sets the threshold for N+1 entries.</p> <p>Values: 0, ..., IC_RESP_BUF_DEPTH</p> <p>Default Value: 1</p> <p>Enabled: IC_DEVICE_ROLE<5</p> <p>Parameter Name: IC_DFLT_RESP_BUF_THLD</p>
Default Command Queue Threshold Value	<p>Reset value of CMD_EMPTY_BUF_THLD field in QUEUE_THLD_CTRL register. CMD_EMPTY_BUF_THLD controls the number of empty locations (or greater) in the Command Queue that trigger CMD_QUEUE_READY_STAT interrupt. The valid range is 0 to IC_CMD_BUF_DEPTH-1. If the programmed value is greater than the maximum suggested value, only the number of bits required to address the full buffer depth is considered. The value of N sets the threshold for N empty locations and a value of 0 sets the threshold to indicate that the queue is completely empty.</p> <p>Values: 0, ..., IC_CMD_BUF_DEPTH</p> <p>Default Value: 1</p> <p>Enabled: IC_DEVICE_ROLE<5</p> <p>Parameter Name: IC_DFLT_CMD_BUF_THLD</p>

Table 3-9 Preset Values Parameters (Continued)

Label	Description
DATA_BUFFER_THLD_CTRL	
Default Receive Start Threshold Value	<p>Reset value of RX_START_THLD field in DATA_BUFFER_THLD_CTRL register.</p> <p>When the controller is set up to initiate a read transfer, it waits until the programmed number of empty locations (or more) are available in its receive buffer before it initiates the read transfer on the I3C Interface. The controller waits for one of the following to be true to initiate the read command.</p> <ol style="list-style-type: none"> 1. The data length number of locations to be empty in the Receive FIFO if command data length is smaller than the start threshold. 2. The threshold number of locations to be empty in the Receive FIFO if command data length is larger than the start threshold <p>The supported values for RX_START_THLD are:</p> <ul style="list-style-type: none"> ■ 000 - 1 ■ 001 - 4 ■ 010 - 8 ■ 011 - 16 ■ 100 - 32 ■ 101 - 64 ■ 110: Reserved ■ 111: Reserved <p>Note: It is recommended to set 'RX_START_THLD' to half of the selected Receive buffer depth (IC_RX_BUF_DEPTH)</p> <p>Values: 0x0, ..., 0x7</p> <p>Default Value: 0x1</p> <p>Enabled: IC_DEVICE_ROLE<5</p> <p>Parameter Name: IC_DFLT_RX_START_THLD</p>

Table 3-9 Preset Values Parameters (Continued)

Label	Description
Default Transmit Start Threshold Value	<p>Reset value of TX_START_THLD field in DATA_BUFFER_THLD_CTRL register. When the controller is set up to initiate a write transfer, it waits until the programmed number of entries (or more) are available in its transmit buffer before it initiates the write transfer on the I3C Interface. The controller waits for one of the following to be true to initiate the write command:</p> <ol style="list-style-type: none"> 1. The data length number of locations are filled in the Transmit FIFO if command data length is smaller than the start threshold. 2. The threshold number of locations are filled in the Transmit FIFO if command data length is larger than the start threshold. <p>The supported values for TX_START_THLD are:</p> <ul style="list-style-type: none"> ■ 000: 1 ■ 001: 4 ■ 010: 8 ■ 011: 16 ■ 100: 32 ■ 101: 64 ■ 110: Reserved ■ 111: Reserved <p>Note: It is recommended to set 'TX_START_THLD' to half of the selected Transmit buffer depth (IC_TX_BUF_DEPTH)</p> <p>Values: 0x0, ..., 0x7</p> <p>Default Value: 0x1</p> <p>Enabled: IC_DEVICE_ROLE<5</p> <p>Parameter Name: IC_DFLT_TX_START_THLD</p>

Table 3-9 Preset Values Parameters (Continued)

Label	Description
Default Receive Buffer Threshold Value	<p>Reset value of RX_BUF_THLD field in DATA_BUFFER_THLD_CTRL register. RX_BUF_THLD controls the number of entries (or greater) in the Receive FIFO that trigger the RX_THLD_STAT interrupt. If the programmed value is greater than the buffer depth, then threshold is set to IC_RX_BUF_DEPTH. The supported values for RX_BUF_THLD are:</p> <ul style="list-style-type: none"> ■ 000: 1 ■ 001: 4 ■ 010: 8 ■ 011: 16 ■ 100: 32 ■ 101: 64 ■ 110: Reserved ■ 111: Reserved <p>Note: It is recommended to set 'RX_BUF_THLD' to half of the selected Receive buffer depth (IC_RX_BUF_DEPTH)</p> <p>Values: 0x0, ..., 0x7</p> <p>Default Value: 0x1</p> <p>Enabled: IC_DEVICE_ROLE<5</p> <p>Parameter Name: IC_DFLT_RX_BUF_THLD</p>
Default Transmit Buffer Threshold Value	<p>Reset value of TX_EMPTY_BUF_THLD in DATA_BUFFER_THLD_CTRL register. TX_EMPTY_BUF_THLD controls the number of empty locations (or greater) in the Transmit FIFO that trigger the TX_THLD_STAT interrupt. If the programmed value is greater than the buffer depth, then the threshold is set to IC_TX_BUF_DEPTH. The supported values for TX_BUF_THLD are:</p> <ul style="list-style-type: none"> ■ 000: 1 ■ 001: 4 ■ 010: 8 ■ 011: 16 ■ 100: 32 ■ 101: 64 ■ 110: Reserved ■ 111: Reserved <p>Note: It is recommended to set 'TX_BUF_THLD' to half of the selected Transmit buffer depth (IC_TX_BUF_DEPTH).</p> <p>Values: 0x0, ..., 0x7</p> <p>Default Value: 0x1</p> <p>Enabled: IC_DEVICE_ROLE<5</p> <p>Parameter Name: IC_DFLT_TX_BUF_THLD</p>

Table 3-9 Preset Values Parameters (Continued)

Label	Description
IBI_QUEUE_CTRL	
Default Value of Notify SIR Rejected	<p>Default value of NOTIFY_SIR_REJECTED field in IBI_QUEUE_CTRL register. In master mode of operation, this bit is used to suppress the reporting of Slave Interrupt requests being rejected to the application.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Disable (0) ■ Enable (1) <p>Default Value: Disable</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DFLT_SIR_REJECTED</p>
Default Value of Notify Master Request Rejected	<p>Default value of NOTIFY_MR_REJECTED field in IBI_QUEUE_CTRL register. In master mode of operation, this bit is used to suppress the reporting of Master requests being rejected to the application.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Disable (0) ■ Enable (1) <p>Default Value: Disable</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DFLT_MR_REJECTED</p>
Default Value of Notify Hot-Join Rejected	<p>Default value of NOTIFY_HJ_REJECTED field in IBI_QUEUE_CTRL register. In master mode of operation, this bit is used to suppress the reporting of Hot-Join requests being rejected to the application.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ Disable (0) ■ Enable (1) <p>Default Value: Disable</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DFLT_HJ_REJECTED</p>
SCL_I3C_OD_TIMING	
Hardware Reset value for I3C Open Drain High Count Register	<p>Reset value of I3C_OD_HCNT field in SCL_HCNT_TIMING register.</p> <p>Values: 0x0, ..., 0xff</p> <p>Default Value: 0xa</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DFLT_I3C_OD_HCNT</p>

Table 3-9 Preset Values Parameters (Continued)

Label	Description
Hardware Reset value for I3C Open Drain Low Count Register	Reset value of I3C_OD_LCNT field in SCL_LCNT_TIMING register. Values: 0x0, ..., 0xff Default Value: 0x10 Enabled: IC_DEVICE_ROLE<4 Parameter Name: IC_DFLT_I3C_OD_LCNT
SCL_I3C_PP_TIMING	
Hardware Reset value for I3C Push Pull High Count Register	Reset value of I3C_PP_HCNT field in SCL_HCNT_TIMING register. Values: 0x0, ..., 0xff Default Value: 0xa Enabled: IC_DEVICE_ROLE<4 Parameter Name: IC_DFLT_I3C_PP_HCNT
SCL_I3C_PP_LCNT	
Hardware Reset value for I3C Push Pull Low Count Register	Reset value of I3C_PP_LCNT field in SCL_LCNT_TIMING register. Values: 0x0, ..., 0xff Default Value: 0xa Enabled: IC_DEVICE_ROLE<4 Parameter Name: IC_DFLT_I3C_PP_LCNT
SCL_I2C_FM_TIMING	
Hardware Reset value for I2C Fast Mode High Count Register	Reset value of I2C_FM_HCNT field in SCL_I2C_FM_TIMING register. Values: 0x0, ..., 0xffff Default Value: 0x10 Enabled: (IC_DEVICE_ROLE<4) && IC_HAS_EXTND_TIMING==1 Parameter Name: IC_DFLT_I2C_FM_HCNT
Hardware Reset value for I2C Fast Mode Low Count Register	Reset value of I2C_FM_LCNT field in SCL_I2C_FM_TIMING register. Values: 0x0, ..., 0xffff Default Value: 0x10 Enabled: (IC_DEVICE_ROLE<4) && IC_HAS_EXTND_TIMING==1 Parameter Name: IC_DFLT_I2C_FM_LCNT
SCL_I2C_FMP_TIMING	
Hardware Reset value for I2C Fast Mode Plus High Count Register	Reset value of I2C_FMP_HCNT field in SCL_I2C_FMP_TIMING register. Values: 0x0, ..., 0xff Default Value: 0x10 Enabled: (IC_DEVICE_ROLE<4) && IC_HAS_EXTND_TIMING==1 Parameter Name: IC_DFLT_I2C_FMP_HCNT
Hardware Reset value for I2C Fast Mode Plus Low Count Register	Reset value of I2C_FMP_LCNT field in SCL_I2C_FMP_TIMING register. Values: 0x0, ..., 0xffff Default Value: 0x10 Enabled: (IC_DEVICE_ROLE<4) && IC_HAS_EXTND_TIMING==1 Parameter Name: IC_DFLT_I2C_FMP_LCNT

Table 3-9 Preset Values Parameters (Continued)

Label	Description
SCL_EXT_LCNT_TIMING	
Hardware Reset value for SCL Extended Low Count 4 Register	<p>Reset value of I3C_EXT_LCNT_4 field in SCL_EXT_LCNT_TIMING register</p> <p>Values: 0x0, ..., 0xff</p> <p>Default Value: 0x20</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DFLT_EXT_LCNT_4</p>
Hardware Reset value for SCL Extended Low Count 3 Register	<p>Reset value of I3C_EXT_LCNT_3 field in SCL_EXT_LCNT_TIMING register</p> <p>Values: 0x0, ..., 0xff</p> <p>Default Value: 0x20</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DFLT_EXT_LCNT_3</p>
Hardware Reset value for SCL Extended Low Count 2 Register	<p>Reset value of I3C_EXT_LCNT_2 field in SCL_EXT_LCNT_TIMING register</p> <p>Values: 0x0, ..., 0xff</p> <p>Default Value: 0x20</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DFLT_EXT_LCNT_2</p>
Hardware Reset value for SCL Extended Low Count 1 Register	<p>Reset value of I3C_EXT_LCNT_1 field in SCL_EXT_LCNT_TIMING register</p> <p>Values: 0x0, ..., 0xff</p> <p>Default Value: 0x20</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DFLT_EXT_LCNT_1</p>
BUS_FREE_AVAIL_TIMING	
Hardware Reset value for Bus Available Time Register	<p>Reset value of BUS_AVAILABLE_TIME field in BUS_FREE_AVAIL_TIMING register</p> <p>Values: 0x0, ..., 0xffff</p> <p>Default Value: 0x20</p> <p>Enabled: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)</p> <p>Parameter Name: IC_DFLT_BUS_AVAIL_CNT</p>
Hardware Reset value for Bus Free Time Register	<p>Reset value of BUS_FREE_TIME field in BUS_FREE_AVAIL_TIMING register</p> <p>Values: 0x0, ..., 0xffff</p> <p>Default Value: 0x20</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DFLT_BUS_FREE_CNT</p>

Table 3-9 Preset Values Parameters (Continued)

Label	Description
BUS_IDLE_TIMING	
Hardware Reset value for Bus Idle Count Register	<p>Reset value of BUS_IDLE field in BUS_IDLE_TIMING register</p> <p>Values: 0x0, ..., 0xfffff</p> <p>Default Value: 0x20</p> <p>Enabled: IC_SLV_HJ==1</p> <p>Parameter Name: IC_DFLT_BUS_IDLE_CNT</p>
SCL_EXT_TERMN_LCNT_TIMING	
Hardware Reset value for HDR Ternary Skew Count Register	<p>Reset value of TS_SKEW_LCNT field in SCL_TERMN_LCNT_TIMING register. Define the skew in number of core clocks which is used for Master Turn-Around Detection. The Specification defines the maximum skew is 12.8 ns. Based on the core_clk period selection, the default value of number of core_clks count is calculated.</p> <p>Values: 0x0, ..., 0xff</p> <p>Default Value: 0x3</p> <p>Enabled: (IC_DEVICE_ROLE<4) && IC_SPEED_HDR_TS==1</p> <p>Parameter Name: IC_DFLT_TS_SKEW_LCNT</p>
Hardware Reset value for I3C Termination Low Count Register	<p>Reset value of TERMIN_LCNT field in SCL_TERMN_LCNT_TIMING register.</p> <p>Values: 0x0, ..., 0xff</p> <p>Default Value: 0x0</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DFLT_TERMN_LCNT</p>
SDA_HOLD_DLY_TIMING	
Hardware Reset value for SDA Transmit Hold Time	<p>Reset value of SDA_TX_HOLD field in SDA_HOLD_SWITCH_DLY_TIMING register.</p> <p>Values: 0x0, ..., 0x7</p> <p>Default Value: 0x1</p> <p>Enabled: IC_DEVICE_ROLE<4</p> <p>Parameter Name: IC_DFLT_SDA_TX_HOLD</p>
SCL_LOW_MST_TIMEOUT_COUNT	
SCL Low Timeout Count	<p>This parameter is used to select the default value of the SCL Low Timeout Count Register.</p> <p>Values: 0x1, ..., 0xffffffff</p> <p>Default Value: 0x3567e0</p> <p>Enabled: IC_HAS_DEVICE_RESET_SUPPORT==1</p> <p>Parameter Name: IC_DFLT_SCL_LOW_TIMEOUT_COUNT</p>

Table 3-9 Preset Values Parameters (Continued)

Label	Description
SLV_TSX_SYMBL_TIMING	
Hardware Reset value for TSX symbol timing Register	<p>Reset value of TSP/TSL symbol count in TSX_SYMBL_TIMING register</p> <p>Values: 0x0, ..., 0x3f</p> <p>Default Value: 0x3f</p> <p>Enabled: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) && IC_SPEED_HDR_TS==1</p> <p>Parameter Name: IC_DFLT_SLV_TSX_SYMBL_CNT</p>

3.10 HCI Preset Values Parameters

Table 3-10 HCI Preset Values Parameters

Label	Description
HCI_VERSION	
Default Value of HCI Version.	<p>Reset value of HCI Version Bit Field in IC_HCI_VERSION register.</p> <p>Values: 0x0, ..., 0xffff</p> <p>Default Value: 0x100</p> <p>Enabled: Always</p> <p>Parameter Name: IC_DFLT_HCI_VERSION</p>
SCL_LCNT_TIMING	
Hardware Reset value for I2C Standard Speed High Count Register	<p>Reset value of I2C_SS_HCNT field in SCL_I2C_SS_TIMING register.</p> <p>Values: 0x0, ..., 0xffff</p> <p>Default Value: 0x10</p> <p>Enabled: (IC_DEVICE_ROLE<4) && IC_HAS_HCI==1</p> <p>Parameter Name: IC_DFLT_I2C_SS_HCNT</p>
Hardware Reset value for I2C Standard Speed Low Count Register	<p>Reset value of I2C_SS_LCNT field in SCL_I2C_SS_TIMING register.</p> <p>Values: 0x0, ..., 0xffff</p> <p>Default Value: 0x10</p> <p>Enabled: (IC_DEVICE_ROLE<4) && IC_HAS_HCI==1</p> <p>Parameter Name: IC_DFLT_I2C_SS_LCNT</p>

4

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.

The I/O signals are grouped as follows:

- APB Interface on [page 189](#)
- RAM Interface on [page 193](#)
- Core Interface on [page 195](#)
- HDR Interface on [page 196](#)
- Interrupt Interface on [page 198](#)
- SDMA Interface on [page 199](#)
- Debug Interface on [page 202](#)
- I3C Interface on [page 204](#)
- Slave Interface on [page 207](#)
- Scan Interface on [page 211](#)

4.1 APB Interface Signals



Table 4-1 APB Interface Signals

Port Name	I/O	Description
pclk	I	<p>APB Interface Clock</p> <p>Exists: (((IC_SLVIF_MODE==1) (IC_SLVIF_MODE==2)) && (IC_DEVICE_ROLE<5)) && (IC_REGIF_MODE==0)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
presetn	I	<p>APB Interface Reset</p> <p>Active low input that asynchronously resets the APB interface to its default state. The reset must be synchronously de-asserted after rising edge of pclk.</p> <p>DWC_MIPI_I3C does not contain logic to perform this synchronization, so it must be provided externally.</p> <p>Exists: (((IC_SLVIF_MODE==1) (IC_SLVIF_MODE==2)) && (IC_DEVICE_ROLE<5)) && (IC_REGIF_MODE==0)</p> <p>Synchronous To: None</p> <p>Registered: N/A</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: Low</p>

Table 4-1 APB Interface Signals (Continued)

Port Name	I/O	Description
psel	I	<p>APB Interface Select Signal Selects the DWC_MIPI_I3C APB slave interface. Must be active for requests to be accepted.</p> <p>Exists: (((IC_SLVIF_MODE==1) (IC_SLVIF_MODE==2)) && (IC_DEVICE_ROLE<5)) && (IC_REGIF_MODE==0)</p> <p>Synchronous To: (IC_MAIN_MASTER_EN==0) && (IC_SLAVE_APB_EN==1) ? "pclk" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==1) ? "pclk" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==0) ? "pclk,core_clk" : (IC_CLK_TYPE==1) ? "pclk,core_clk" : "pclk"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
penable	I	<p>APB Interface Enable Signal Asserted for a single pclk cycle and used for timing read/write operations.</p> <p>Exists: (((IC_SLVIF_MODE==1) (IC_SLVIF_MODE==2)) && (IC_DEVICE_ROLE<5)) && (IC_REGIF_MODE==0)</p> <p>Synchronous To: (IC_MAIN_MASTER_EN==0) && (IC_SLAVE_APB_EN==1) ? "pclk" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==1) ? "pclk" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==0) ? "pclk,core_clk" : (IC_CLK_TYPE==1) ? "pclk,core_clk" : "pclk"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
pwrite	I	<p>APB Interface Write Signal Asserted for a single pclk cycle and used for timing read/write operations.</p> <p>Exists: (((IC_SLVIF_MODE==1) (IC_SLVIF_MODE==2)) && (IC_DEVICE_ROLE<5)) && (IC_REGIF_MODE==0)</p> <p>Synchronous To: (IC_MAIN_MASTER_EN==0) && (IC_SLAVE_APB_EN==1) ? "pclk" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==1) ? "pclk" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==0) ? "pclk,core_clk" : (IC_CLK_TYPE==1) ? "pclk,core_clk" : "pclk"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>

Table 4-1 APB Interface Signals (Continued)

Port Name	I/O	Description
paddr[(IC_SLVIF_ADDR_WIDTH-1):0]	I	<p>APB Interface Address Bus Uses lower 12 bits of the address bus for register decode.</p> <p>Exists: (((IC_SLVIF_MODE==1) (IC_SLVIF_MODE==2)) && (IC_DEVICE_ROLE<5)) && (IC_REGIF_MODE==0)</p> <p>Synchronous To: (IC_MAIN_MASTER_EN==0) && (IC_SLAVE_APB_EN==1) ? "0:1=None;31:2=pclk" : (IC_MAIN_MASTER_EN==1) && (IC_SLAVE_LITE_ONLY_EN==0) && (IC_HAS_HCI==0) ? "0:1=None;31:2=pclk,core_clk" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==0) ? "pclk,core_clk" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==1) && (IC_HAS_HCI==0) ? "None" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==1) && (IC_HAS_HCI==1) && (IC_HAS_DAT==0) ? "None" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==1) && (IC_HAS_HCI==1) && (IC_HAS_DAT==1) ? "0:1=None;2:6=pclk;7:11=None;31:12=pclk" : (IC_CLK_TYPE==1) ? "pclk,core_clk" : "pclk"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
pwdata[(IC_SLVIF_DATA_WIDTH-1):0]	I	<p>APB Interface Write Data Bus Driven by the bus master (DW_ahb to DW_apb bridge or DW_axi to DW_axi_x2p bridge) during write cycles.</p> <p>Exists: (((IC_SLVIF_MODE==1) (IC_SLVIF_MODE==2)) && (IC_DEVICE_ROLE<5)) && (IC_REGIF_MODE==0)</p> <p>Synchronous To: (IC_MAIN_MASTER_EN==0) && (IC_SLAVE_APB_EN==1) ? "pclk" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==1) ? "None" : (IC_SLAVE_LITE_ONLY_EN==0) && (IC_CLK_TYPE==0) ? "pclk,core_clk" : (IC_CLK_TYPE==1) ? "pclk,core_clk" : "pclk"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
pready	O	<p>APB Interface Ready Signal Indicates whether a request cycle is accepted or not.</p> <p>Exists: (((IC_SLVIF_MODE==1) (IC_SLVIF_MODE==2)) && (IC_DEVICE_ROLE<5)) && (IC_REGIF_MODE==0)</p> <p>Synchronous To: pclk</p> <p>Registered: Yes</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>

Table 4-1 APB Interface Signals (Continued)

Port Name	I/O	Description
prdata[(IC_SLVIF_DATA_WIDTH-1):0]	O	<p>APB Interface Read Data Bus Driven by the DWC_mipi_i3c controller during read cycles.</p> <p>Exists: (((IC_SLVIF_MODE==1) (IC_SLVIF_MODE==2)) && (IC_DEVICE_ROLE<5)) && (IC_REGIF_MODE==0)</p> <p>Synchronous To: pclk Registered: Yes Power Domain: SINGLE_DOMAIN Active State: High</p>
pslverr	O	<p>APB Interface Slave Error Response Signal DWC_mipi_i3c controller does not use this signal, and is kept for consistency purpose.</p> <p>Exists: (((IC_SLVIF_MODE==1) (IC_SLVIF_MODE==2)) && (IC_DEVICE_ROLE<5)) && (IC_REGIF_MODE==0)</p> <p>Synchronous To: (IC_REGIF_MODE_EN==0) ? "None":pclk Registered: No Power Domain: SINGLE_DOMAIN Active State: High</p>

4.2 RAM Interface Signals



Table 4-2 RAM Interface Signals

Port Name	I/O	Description
ram_cs_n	O	<p>SPRAM Chip Select</p> <ul style="list-style-type: none"> ■ 0 - SPRAM is selected ■ 1 - SPRAM is not selected <p>Exists: (IC_DEVICE_ROLE<5) Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "core_clk" Registered: (IC_FW_RAM_RETIMING_EN==1) ? "Yes" : "No" Power Domain: SINGLE_DOMAIN Active State: Low</p>
ram_wr_n	O	<p>SPRAM Read/Write Control</p> <ul style="list-style-type: none"> ■ 0 - Write enable ■ 1 - Read enable <p>Exists: (IC_DEVICE_ROLE<5) Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "core_clk" Registered: (IC_FW_RAM_RETIMING_EN==1) ? "Yes" : "No" Power Domain: SINGLE_DOMAIN Active State: Low</p>
ram_addr[(IC_RAM_ADDR_WIDTH-1):0]	O	<p>SPRAM Address</p> <p>Exists: (IC_DEVICE_ROLE<5) Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "core_clk" Registered: (IC_FW_RAM_RETIMING_EN==1) ? "Yes" : "No" Power Domain: SINGLE_DOMAIN Active State: High</p>

Table 4-2 RAM Interface Signals (Continued)

Port Name	I/O	Description
ram_data_in[(IC_RAM_DATA_WIDTH-1):0]	I	<p>SPRAM Data Input</p> <p>Exists: (IC_DEVICE_ROLE<5)</p> <p>Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "core_clk"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
ram_data_out[(IC_RAM_DATA_WIDTH-1):0]	O	<p>SPRAM Data Output</p> <p>Exists: (IC_DEVICE_ROLE<5)</p> <p>Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "core_clk"</p> <p>Registered: (IC_FW_RAM_RETIMING_EN==1) ? "Yes" : "No"</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>

4.3 Core Interface Signals



Table 4-3 Core Interface Signals

Port Name	I/O	Description
core_clk	I	<p>Core Interface Clock DWC_mipi_i3c controller interface clock</p> <p>Exists: (IC_DEVICE_ROLE<4)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
core_rst_n	I	<p>Core Interface Reset Active low input that asynchronously resets the Core interface to its default state. The reset must be synchronously de-asserted after the rising edge of the <code>core_clk</code>. DWC_mipi_i3c does not contain a logic to perform this synchronization, so it must be provided externally.</p> <p>Exists: (IC_DEVICE_ROLE<4)</p> <p>Synchronous To: None</p> <p>Registered: N/A</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: Low</p>

4.4 HDR Interface Signals

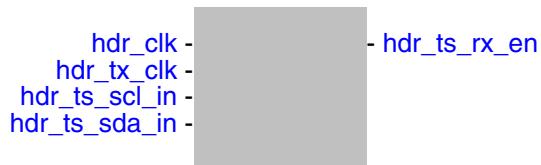


Table 4-4 HDR Interface Signals

Port Name	I/O	Description
hdr_clk	I	<p>HDR-TS Recovered Clock HDR clock generated from the incoming ternary symbols through external clock recovery block (CDR).</p> <p>Exists: (IC_SPEED_HDR_TS==1) && (IC_DEVICE_ROLE<5)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
hdr_tx_clk	I	<p>HDR-TS Transmit Clock HDR Transmit clock required for generation of the ternary symbols for HDR-TS read transfers in Slave mode of operation.</p> <p>Exists: (IC_SPEED_HDR_TS==1) && (IC_DEVICE_ROLE>1)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
hdr_ts_scl_in	I	<p>Incoming HDR SCL Line Recovered SCL signal for HDR-TS mode. This SCL is recovered from the CDR block outside of DWC_mipi_i3c.</p> <p>Exists: (IC_SPEED_HDR_TS==1) && (IC_DEVICE_ROLE<5)</p> <p>Synchronous To: hdr_clk</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>

Table 4-4 HDR Interface Signals (Continued)

Port Name	I/O	Description
hdr_ts_sda_in	I	<p>Incoming HDR SDA Line Recovered SDA signal for HDR-TS mode. This SDA is recovered from the CDR block outside of DWC_mipi_i3c.</p> <p>Exists: (IC_SPEED_HDR_TS==1) && (IC_DEVICE_ROLE<5) Synchronous To: hdr_clk Registered: No Power Domain: SINGLE_DOMAIN Active State: High</p>
hdr_ts_rx_en	O	<p>HDR TS clock Recovery Block Enable This signal is used to enable the external clock recovery block for HDR Ternary Mode.</p> <p>Exists: (IC_SPEED_HDR_TS==1) && (IC_DEVICE_ROLE<5) Synchronous To: (IC_MAIN_MASTER_EN==1) && (IC_SLAVE_APB_EN==1) ? "scl_in_a_n" : (IC_MAIN_MASTER_EN==1) && (IC_SLAVE_APB_EN==0) ? "None" : (IC_MAIN_MASTER_EN==0) && (IC_SLAVE_APB_EN==1) ? "scl_in_a_n" : "None" Registered: (IC_MAIN_MASTER_EN==1) && (IC_SLAVE_APB_EN==1) ? "No" : "Yes" Power Domain: SINGLE_DOMAIN Active State: High</p>

4.5 Interrupt Interface Signals



Table 4-5 Interrupt Interface Signals

Port Name	I/O	Description
ic_intr	O	<p>Combined Active High interrupt</p> <p>Exists: (IC_INTR_IO==1) && (IC_INTR_POL==1) && (IC_DEVICE_ROLE<5)</p> <p>Synchronous To: pclk</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>

4.6 SDMA Interface Signals



Table 4-6 SDMA Interface Signals

Port Name	I/O	Description
dma_ack_tx	I	<p>DMA transmit acknowledgement sent by the DMA Controller to acknowledge the end of each DMA burst or single transaction to the transmit FIFO.</p> <p>Exists: (IC_HAS_DMA==1) && (IC_DEVICE_ROLE<5)</p> <p>Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "dma_clk"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
dma_ack_rx	I	<p>DMA receive acknowledgement sent by the DMA controller to acknowledge the end of each DMA burst or single transaction from the receive FIFO.</p> <p>Exists: (IC_HAS_DMA==1) && (IC_DEVICE_ROLE<5)</p> <p>Synchronous To: (IC_MAIN_MASTER_EN==1) && (IC_SLAVE_APB_EN==1) && (IC_DMA_CORE_CLK_ASYNC==0) ? "core_clk,dma_clk,pclk" : (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "dma_clk"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
dma_req_tx	O	<p>Transmit FIFO DMA Request.</p> <p>This signal is asserted when the transmit FIFO requires service from the DMA controller, that is, the transmit FIFO is at or below the threshold level.</p> <p>Exists: (IC_HAS_DMA==1) && (IC_DEVICE_ROLE<5)</p> <p>Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "dma_clk"</p> <p>Registered: Yes</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>

Table 4-6 SDMA Interface Signals (Continued)

Port Name	I/O	Description
dma_req_rx	O	<p>Receive FIFO DMA Request Asserted when the receive FIFO requires service from the DMA Controller, that is, the receive FIFO is at or above the threshold level.</p> <p>Exists: (IC_HAS_DMA==1) && (IC_DEVICE_ROLE<5) Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "dma_clk" Registered: Yes Power Domain: SINGLE_DOMAIN Active State: High</p>
dma_single_tx	O	<p>DMA Transmit FIFO Single Signal This DMA status output informs the DMA Controller that there is at least one free entry in the transmit FIFO.</p> <p>Exists: (IC_HAS_DMA==1) && (IC_DEVICE_ROLE<5) Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "dma_clk" Registered: Yes Power Domain: SINGLE_DOMAIN Active State: High</p>
dma_single_rx	O	<p>DMA Receive FIFO Single Signal This DMA status output informs the DMA Controller that there is at least one valid data entry in the receive FIFO.</p> <p>Exists: (IC_HAS_DMA==1) && (IC_DEVICE_ROLE<5) Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "dma_clk" Registered: Yes Power Domain: SINGLE_DOMAIN Active State: High</p>
dma_last_rx	O	<p>Indicates last data to the external dma for rx channel. Applicable in only Slave mode of operation.</p> <p>Exists: (IC_HAS_DMA==1) && (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "dma_clk" Registered: Yes Power Domain: SINGLE_DOMAIN Active State: High</p>

Table 4-6 SDMA Interface Signals (Continued)

Port Name	I/O	Description
dma_clk	I	<p>SDMA Interface clock DWC_mipi_i3c SDMA interface clock</p> <p>Exists: (IC_HAS_DMA==1) && (IC_DEVICE_ROLE<4)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
dma_rst_n	I	<p>SDMA Interface Reset</p> <p>Active low input that asynchronously resets the SDMA interface to its default state. The reset must be synchronously de-asserted after the rising edge of dma_clk.</p> <p>DWC_mipi_i3c does not contain a logic to perform this synchronization, so it must be provided externally.</p> <p>Exists: (IC_HAS_DMA==1) && (IC_DEVICE_ROLE<4)</p> <p>Synchronous To: None</p> <p>Registered: N/A</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: Low</p>

4.7 Debug Interface Signals



Table 4-7 Debug Interface Signals

Port Name	I/O	Description
debug_port[(IC_DBG_DW-1):0]	O	<p>Debug port On chip debug signal as follows:</p> <ul style="list-style-type: none"> ■ Master Mode Debug Signals <p>debug[0] = Indicates if Master is the current master or not. debug[1] = Indicates if SCL is in Push-pull or Open-Drain Mode. debug[2] = Indicates if SDA is in Push-Pull or Open-Drain Mode. debug[3] = Indicates SCL signal level. debug[4] = Indicates SDA signal level. debug[5] = Indicates if Command Queue is full. debug[6] = Indicates if Command Queue is empty. debug[7] = Indicates if Response Queue is full. debug[8] = Indicates if Response Queue is empty. debug[9] = Indicates if IBI Queue is full. debug[10] = Indicates if IBI Queue is empty. debug[11] = Indicates if Transmit Buffer is full. debug[12] = Indicates if Transmit Buffer is empty. debug[13] = Indicates Receive Buffer full. debug[14] = Indicates Receive Buffer empty. debug[20:15] = Indicates Current Master Transfer command. debug[26:21] = Indicates Current Master Transfer command state. debug[30:27] = Indicates Current Master Transfer command TID. debug[31] = Indicates if the Master controller is in Idle state. debug[32] = Indicates the Enable status of the controller.</p> <p>debug[33] = Indicates if Periodic Poll command is in progress. debug[34] = Indicates Periodic Poll slot tick change pulse. debug[47:35] = Reserved</p> <p>debug[47:33] = Reserved</p>

Table 4-7 Debug Interface Signals (Continued)

Port Name	I/O	Description
debug_port[(IC_DBG_DW-1):0] (Cont.)	O	<ul style="list-style-type: none"> ■ Slave Mode Debug Signals <p>debug[4:0] = Reserved debug[5] = Indicates if Command Queue is full. debug[6] = Indicates if Command Queue is empty. debug[7] = Indicates if Response Queue is full. debug[8] = Indicates if Response Queue is empty. debug[10:9] = Reserved debug[11] = Indicates if Transmit Buffer is full. debug[12] = Indicates if Transmit Buffer is empty. debug[13] = Indicates if Receive Buffer is full. debug[14] = Indicates if Receive Buffer is empty. debug[31:15] = Reserved debug[32] = Indicates the Enable status of the controller. debug[34:33] = Reserved debug[39:35] = Indicates Slave Transfer State. debug[43:40] = Indicates Slave Command TID. debug[47:44] = Reserved</p> <p>Exists: (IC_HAS_DEBUG_PORTS==1) Synchronous To: (IC_DEVICE_ROLE==5) ? "scl_in_a" : ((IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : "core_clk") Registered: No Power Domain: SINGLE_DOMAIN Active State: High</p>

4.8 I3C Interface Signals

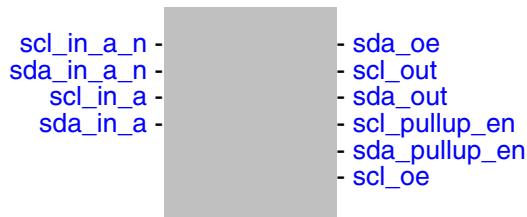


Table 4-8 I3C Interface Signals

Port Name	I/O	Description
scl_in_a_n	I	<p>Incoming inverted SCL Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) ((IC_DEVICE_ROLE==1) && (IC_SPEED_HDR_DDR==1) && (IC_SDA_SAMPLING_IN_SCL==1))</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
sda_in_a_n	I	<p>Incoming Inverted SDA Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1)</p> <p>Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_SPEED_HDR_EN==1) ? "scl_in_a,scl_in_a_n" : (IC_SLAVE_APB_EN==1) && (IC_SPEED_HDR_EN==0) ? "scl_in_a" : "None"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
sda_oe	O	<p>Outgoing SDA Pad enable This signal is used to enable the pad to switch between OD(0) and PP(1) mode.</p> <p>Exists: Always</p> <p>Synchronous To: ((IC_DEVICE_ROLE<4) && (IC_DEVICE_ROLE>1)) ? "core_clk,scl_in_a,scl_in_a_n" : ((IC_DEVICE_ROLE>3) ? "scl_in_a,scl_in_a_n" : "core_clk")</p> <p>Registered: (IC_DEVICE_ROLE>1) ? "No" : "Yes"</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>

Table 4-8 I3C Interface Signals (Continued)

Port Name	I/O	Description
scl_out	O	<p>SCL output Signal This signal is an input to the SCL PAD Driver.</p> <p>Exists: ((IC_OPEN_DRAIN_CLASS_PULLUP==1) && (IC_DEVICE_ROLE<4)) ((IC_DEVICE_ROLE>1) && (IC_SPEED_HDR_TS==1))</p> <p>Synchronous To: (((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<4)) && (IC_SPEED_HDR_TS==1)) ? "core_clk, hdr_tx_clk" : ((IC_DEVICE_ROLE==4) && (IC_SPEED_HDR_TS==1)) ? "hdr_tx_clk" : "core_clk"</p> <p>Registered: ((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<4)) ? "No" : "Yes"</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
sda_out	O	<p>SDA output Signal This signal is an input to the SDA PAD Driver.</p> <p>Exists: (IC_OPEN_DRAIN_CLASS_PULLUP==1) (IC_DEVICE_ROLE==5)</p> <p>Synchronous To: ((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<4)) ? "core_clk, scl_in_a, scl_in_a_n" : ((IC_DEVICE_ROLE>3) ? "scl_in_a, scl_in_a_n" : "core_clk")</p> <p>Registered: (IC_DEVICE_ROLE>1) ? "No" : "Yes"</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
scl_pullup_en	O	<p>SCL Open-Drain Pull-Up Signal Applicable only in Master mode of operation. This signal enables the Pull-Up resistor or equivalent current source in Open-Drain mode when the controller requires to drive '1' on the SCL line.</p> <p>Exists: (IC_OPEN_DRAIN_CLASS_PULLUP==1) && (IC_DEVICE_ROLE<4)</p> <p>Synchronous To: None</p> <p>Registered: Yes</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
sda_pullup_en	O	<p>SDA Open-Drain Pull-Up Signal Applicable in only Master mode of operation. This signal enables the Pull-Up resistor or equivalent current source in Open-Drain mode when the controller requires to drive '1' on the SDA line.</p> <p>Exists: (IC_OPEN_DRAIN_CLASS_PULLUP==1) && (IC_DEVICE_ROLE<4)</p> <p>Synchronous To: core_clk</p> <p>Registered: Yes</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>

Table 4-8 I3C Interface Signals (Continued)

Port Name	I/O	Description
scl_oe	O	<p>Outgoing SCL Pad enable This signal is used to enable the Pad to switch between OD(0) and PP(1) mode.</p> <p>Exists: (IC_DEVICE_ROLE<4) (IC_SPEED_HDR_TS==1)</p> <p>Synchronous To: (((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<4)) && (IC_SPEED_HDR_TS==1)) ? "core_clk,hdr_tx_clk" : ((IC_DEVICE_ROLE==4) && (IC_SPEED_HDR_TS==1)) ? "hdr_tx_clk" : ((IC_DEVICE_ROLE==1) && (IC_SPEED_HDR_TS==1)) && (IC_OPEN_DRAIN_CLASS_PULLUP_EN==1)) ? "core_clk" : ((IC_DEVICE_ROLE==1) && (IC_SPEED_HDR_TS==0) && (IC_OPEN_DRAIN_CLASS_PULLUP_EN==0)) ? "core_clk" : "None"</p> <p>Registered: ((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<4)) ? "No" : "Yes"</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
scl_in_a	I	<p>Incoming SCL Input SCL Signal from the SCL Pad Driver.</p> <p>Exists: Always</p> <p>Synchronous To: (IC_DEVICE_ROLE==1) ? "core_clk" : "None"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
sda_in_a	I	<p>Incoming SDA Input SDA Signal from the SDA Pad Driver.</p> <p>Exists: Always</p> <p>Synchronous To: (IC_DEVICE_ROLE==4) && (IC_SPEED_HDR_TS==1) ? "hdr_tx_clk,pclk,scl_in_a,scl_in_a_n" : ((IC_DEVICE_ROLE==4) && (IC_SPEED_HDR_TS==0)) ? "pclk,scl_in_a,scl_in_a_n" : "None"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>

4.9 Slave Interface Signals

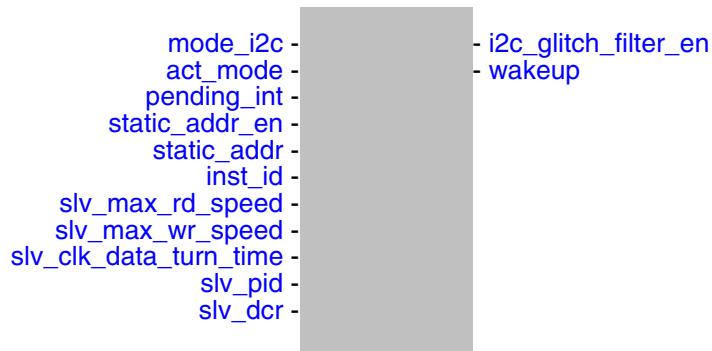


Table 4-9 Slave Interface Signals

Port Name	I/O	Description
mode_i2c	I	<p>I2C or I3C mode select signal Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) Synchronous To: None Registered: No Power Domain: SINGLE_DOMAIN Active State: Not applicable</p>
act_mode[1:0]	I	<p>Slave activity mode for GETSTATUS CCC Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : (IC_SLAVE_LITE_ONLY_EN==1) ? "scl_in_a": "core_clk" Registered: (IC_DEVICE_ROLE==5) ? "Yes" : "No" Power Domain: SINGLE_DOMAIN Active State: High</p>
pending_int[3:0]	I	<p>Pending interrupt information for GETSTATUS CCC Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : (IC_SLAVE_LITE_ONLY_EN==1) ? "scl_in_a": "core_clk" Registered: (IC_DEVICE_ROLE==5) ? "Yes" : "No" Power Domain: SINGLE_DOMAIN Active State: High</p>

Table 4-9 Slave Interface Signals (Continued)

Port Name	I/O	Description
static_addr_en	I	<p>Slave static address valid Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) Synchronous To: None Registered: No Power Domain: SINGLE_DOMAIN Active State: High</p>
static_addr[6:0]	I	<p>Slave static address Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) Synchronous To: None Registered: No Power Domain: SINGLE_DOMAIN Active State: High</p>
inst_id[3:0]	I	<p>Slave instance ID Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_SLV_UNIQUE_ID_PROG==0) Synchronous To: None Registered: No Power Domain: SINGLE_DOMAIN Active State: High</p>
i2c_glitch_filter_en	O	<p>I2C 50ns glitch filter enable Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) Synchronous To: (IC_SLAVE_APB_EN==1) && (IC_MAIN_MASTER_EN==0) ? "pclk" : (IC_DEVICE_ROLE==5) ? "scl_in_a" : "core_clk" Registered: Yes Power Domain: SINGLE_DOMAIN Active State: High</p>
wakeup	O	<p>Slave wakeup signal Applicable only in Slave mode of operation.</p> <p>Exists: IC_DEVICE_ROLE>1 Synchronous To: scl_in_a Registered: (IC_SPEED_HDR_TS_EN==1) ? "No" : "Yes" Power Domain: SINGLE_DOMAIN Active State: High</p>

Table 4-9 Slave Interface Signals (Continued)

Port Name	I/O	Description
slv_max_rd_speed[2:0]	I	<p>Slave maximum read data rate Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) && (IC_SLV_MXDS_PROG==1)</p> <p>Synchronous To: (IC_DEVICE_ROLE==4) ? "pclk" : "core_clk"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
slv_max_wr_speed[2:0]	I	<p>Slave maximum write data rate Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) && (IC_SLV_MXDS_PROG==1)</p> <p>Synchronous To: (IC_DEVICE_ROLE==4) ? "pclk" : "core_clk"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
slv_clk_data_turn_time[2:0]	I	<p>Slave maximum clock data turnaround time Applicable only in Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) && (IC_SLV_MXDS_PROG==1)</p> <p>Synchronous To: (IC_DEVICE_ROLE==4) ? "pclk" : "core_clk"</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>
slv_pid[47:0]	I	<p>Slave Device 48-bit Provisioned ID Applicable in only Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_SLV_UNIQUE_ID_PROG==1)</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: High</p>

Table 4-9 Slave Interface Signals (Continued)

Port Name	I/O	Description
slv_dcr[7:0]	I	<p>Device Characteristic Register value Applicable in only Slave mode of operation.</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_SLV_UNIQUE_ID_PROG==1) Synchronous To: None Registered: No Power Domain: SINGLE_DOMAIN Active State: High</p>

4.10 Scan Interface Signals

slv_test_mode -



Table 4-10 Scan Interface Signals

Port Name	I/O	Description
slv_test_mode	I	<p>Test Mode Used to ensure that test automation tools can control all asynchronous flip-flop signals. During scan, set this signal high all the time. In normal operation, tie this signal low.</p> <p>Exists: (IC_DEVICE_ROLE>1) ((IC_DEVICE_ROLE==1) && (IC_SPEED_HDR_DDR==1) && (IC_SDA_SAMPLING_IN_SCL==1))</p> <p>Synchronous To: None</p> <p>Registered: No</p> <p>Power Domain: SINGLE_DOMAIN</p> <p>Active State: Low</p>

5

Register Descriptions

This chapter details all possible registers in the controller. They are arranged hierarchically into maps and blocks (banks). For configurable IP titles, 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 5-1 Possible Read and Write Behaviors

Read (or Write) Behavior	Description
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 to this register once 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 5-2 Memory Access Examples

Memory Access	Description
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 5-3 Optional Attributes

Attribute	Description
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.

Component Memory Maps

The following table documents the memory maps visible in this component.

Table 5-4 DWC_mipi_i3c Memory Maps

Memory Map	Description
DWC_mipi_i3c_HCI_map on page 216	DWC MIPI I3C HCI Memory Map Exists: IC_HAS_HCI==1
DWC_mipi_i3c_map on page 216	DWC MIPI I3C Memory Map Exists: IC_HAS_HCI==0

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 5-5 Address Banks/Blocks for Memory Map: DWC_mipi_i3c_HCI_map

Address Block	Description
DWC_mipi_i3c_HCI_block on page 217	DWC_mipi_i3c_HCI Exists: Always

Table 5-6 Address Banks/Blocks for Memory Map: DWC_mipi_i3c_map

Address Block	Description
DWC_mipi_i3c_block on page 310	DWC_mipi_i3c Exists: Always

5.1 DWC_mipi_i3c_HCI_map/DWC_mipi_i3c_HCI_block Registers

DWC_mipi_i3c_HCI . Follow the link for the register to see a detailed description of the register.

Table 5-7 Registers for Address Block: DWC_mipi_i3c_HCI_map/DWC_mipi_i3c_HCI_block

Register	Offset	Description
HCI_VERSION on page 220	0x0	HCI Version register
HC_CONTROL on page 221	0x4	HC_CONTROL
MASTER_DEVICE_ADDR on page 225	0x8	Master Device Address Register
HC_CAPABILITIES on page 226	0xc	HC Capabilities Register
RESET_CONTROL on page 228	0x10	Reset Control Register
PRESENT_STATE on page 230	0x14	Present State Register
INTR_STATUS on page 231	0x20	Interrupt Status Register
INTR_STATUS_ENABLE on page 232	0x24	Interrupt Status Enable Register
INTR_SIGNAL_ENABLE on page 233	0x28	Interrupt Signal Enable Register
INTR_FORCE on page 234	0x2c	Interrupt Force Enable Register
DAT_SECTION_OFFSET on page 235	0x30	Device Address Table Pointer
DCT_SECTION_OFFSET on page 236	0x34	Device Characteristics Table Pointer
RING_HEADERS_SECTION_OFFSET on page 238	0x38	Ring Headers Section Offset Register
PIO_SECTION_OFFSET on page 239	0x3c	PIO Section Offset Register
EXTCAPS_SECTION_OFFSET on page 240	0x40	Extended Capabilities Section Offset Register
IBI_NOTIFY_CTRL on page 241	0x58	IBI Queue Control Register
DEV_CTX_BASE_LO on page 243	0x60	Device Context Address Low Register
DEV_CTX_BASE_HI on page 244	0x64	Device Context Address High Register
HW_IDENTIFICATION_HEADER on page 245	* Varies	Hardware Identification header Register
COMP_MANUFACTURER on page 246	* Varies	DWC_mipi_i3c MIPI Manufacturer ID Register
COMP_VERSION on page 247	* Varies	DWC_mipi_i3c Version ID Register
COMP_TYPE on page 248	* Varies	DWC_mipi_i3c Version Type Register
BUS_TIMING_HEADER on page 249	* Varies	Bus Timing Header Register
SCL_I3C_OD_TIMING on page 250	* Varies	SCL I3C Open Drain Timing Register

Table 5-7 Registers for Address Block: DWC_mipi_i3c_HCI_map/DWC_mipi_i3c_HCI_block (Continued)

Register	Offset	Description
SCL_I3C_PP_TIMING on page 251	* Varies	SCL I3C Push Pull Timing Register
SCL_I2C_FM_TIMING on page 252	* Varies	SCL I2C Fast Mode Timing Register
SCL_I2C_FMP_TIMING on page 253	* Varies	SCL I2C Fast Mode Plus Timing Register
SCL_I2C_SS_TIMING on page 254	* Varies	SCL I2C Standard Speed Timing Register
SCL_EXT_LCNT_TIMING on page 255	* Varies	SCL Extended Low Count Timing Register
SCL_EXT_TERMN_LCNT_TIMING on page 257	* Varies	SCL Termination Bit Low count Timing Register
SDA_HOLD_SWITCH_DLY_TIMING on page 259	* Varies	SDA Hold and Mode Switch Delay Timing Register
BUS_FREE_TIMING on page 261	* Varies	Bus Free Timing Register
SCL_LOW_MST_EXT_TIMEOUT on page 262	* Varies	SCL_LOW_MST_EXT_TIMEOUT
DS_EXTCAP_HEADER on page 263	* Varies	Debug specific header Register
QUEUE_STATUS_LEVEL on page 264	* Varies	Queue Status Level Register
DATA_BUFFER_STATUS_LEVEL on page 266	* Varies	Data Buffer Status Level Register
PRESENT_STATE_DEBUG on page 267	* Varies	Present State Register
MASTER_EXT_HEADER on page 273	* Varies	Master Extended specific header Register
MASTER_CONFIG on page 274	* Varies	Master Config Register
COMMAND_QUEUE_PORT on page 276	* Varies	COMMAND_QUEUE_PORT
RESPONSE_QUEUE_PORT on page 277	* Varies	Response Queue Port Register
RX_DATA_PORT on page 280	* Varies	Receive Data Port Register
TX_DATA_PORT on page 281	* Varies	Transmit Data Port Register
IBI_PORT on page 282	* Varies	IBI Port Register
QUEUE_THLD_CTRL on page 283	* Varies	Queue Threshold Control Register
DATA_BUFFER_THLD_CTRL on page 286	* Varies	Data Buffer Threshold Control Register
QUEUE_SIZE_CTRL on page 291	* Varies	Queue Size Register
PIO_INTR_STATUS on page 293	* Varies	PIO Interrupt Status Register
PIO_INTR_STATUS_ENABLE on page 295	* Varies	PIO Interrupt Status Enable Register

Table 5-7 Registers for Address Block: DWC_mipi_i3c_HCI_map/DWC_mipi_i3c_HCI_block (Continued)

Register	Offset	Description
PIO_INTR_SIGNAL_ENABLE on page 297	* Varies	PIO Interrupt Signal Enable Register
PIO_INTR_FORCE on page 299	* Varies	PIO Interrupt Force Register
DEV_ADDR_TABLE1_LOC1 on page 301	* Varies	Device Address Table Location 1 of Device1
DEV_ADDR_TABLE1_LOC2 on page 304	* Varies	Device Address Table Location 2 of Device1
DEV_CHAR_TABLE1_LOC1 on page 306	* Varies	Device Characteristic Table Location-1 of Device1 1
DEV_CHAR_TABLE1_LOC2 on page 307	* Varies	Device Characteristic Table Location-2 of Device1 1
DEV_CHAR_TABLE1_LOC3 on page 308	* Varies	Device Characteristic Table Location-3 of Device1 1
DEV_CHAR_TABLE1_LOC4 on page 309	* Varies	Device Characteristic Table Location-4 of Device1 1

5.1.1 HCI_VERSION

- **Name:** HCI Version register
- **Description:** This register indicates the version number of the I3C HCI Specification (this Specification) that the Host Controller implements. The definition of this register does not change across all versions of the I3C HCI Specification.
- **Size:** 32 bits
- **Offset:** 0x0
- **Exists:** IC_DEVICE_ROLE<4



Table 5-8 Fields for Register: HCI_VERSION

Bits	Name	Memory Access	Description
31:0	VERSION	R	<p>HCI Version Major/Minor version of I3C HCI that the Host Controller implements. Lowest 8 bits encode the minor number and the revision. Each 4 bits in this field are encoded as binary encoded decimals. Examples: 0x00000052: HCI v0.5r02 0x00000100: HCI v1.0</p> <p>Value After Reset: IC_DFLT_HCI_VERSION</p> <p>Exists: Always</p>

5.1.2 HC_CONTROL

- **Name:** HC_CONTROL
- **Description:** Host Controller Control register is used to manage the Host Controller and Master Configuration.
- **Size:** 32 bits
- **Offset:** 0x4
- **Exists:** (IC_DEVICE_ROLE<5)

BUS_ENABLE	31
RESUME	30
ABORT	29
Rsvd	28:9
HOT_JOIN_CTRL	8
I2C_SLAVE_PRESENT	7
Rsvd	6:1
IBA_INCLUDE	0

Table 5-9 Fields for Register: HC_CONTROL

Bits	Name	Memory Access	Description
31	BUS_ENABLE	R/W	<p>Host Controller Bus Enable Enables or disables the operation on the I3C Bus by the Host Controller. If the software sets this bit, then it also confirms that initialization is done, and that the Host Controller can use the programmed register values (For example, generation of SCL on IBI detection, and so on). If this bit is not set, then the Host Controller does not generate SCL for incoming IBI. Software may disable the Host Controller bus operation while it is active, However: 1) If a disable request occurs while receiving IBI, the actual disabling does not occur until reception of the IBI is complete. 2) If a disable request occurs while the Host Controller still has additional Commands queued for transfers, then the actual disabling does not occur until transmission of all queued Commands is complete (that is, until the Host Controller encounters a Command with TOC set to 1). The Host Controller then returns to the Idle state (as indicated in register PRESENT_STATE). When the software reads the value 1'b0 from this field, this indicates that the Host Controller bus operation disable operation has completed.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): The Host Controller bus operation is Disabled ■ 0x1 (ENABLED): The Host Controller bus operation is Enabled <p>Value After Reset: 0x0</p> <p>Exists: Always</p> <p>Volatile: true</p>

Table 5-9 Fields for Register: HC_CONTROL (Continued)

Bits	Name	Memory Access	Description
30	RESUME	R/W	<p>Host Controller Resume. This bit is used to resume Host Controller operation following the Halt state. The Host Controller enters the Halt state (as indicated in register PRESENT_STATE) as a result of any type of error occurring in a transfer. The error type is indicated by the field ERR_STATUS in register RESPONSE_QUEUE_PORT). After the Host Controller has entered the Halt state, the application must write the value 1'b1 to the RESUME bit to resume Host Controller operation. The Host Controller automatically clears the RESUME bit once it has resumed making transfers, that is, has initiated the next Command.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (RUNNING): The Host Controller Running ■ 0x1 (SUSPENDED): The Host Controller is Suspended(RW1C) <p>Value After Reset: 0x0</p> <p>Exists: Always</p> <p>Volatile: true</p>
29	ABORT	R/W	<p>Host Controller Abort. When set to 1, this bit allows the Host Controller to relinquish control of the I3C Bus before completing the currently issued transfer. In response to an ABORT request, the Host Controller issues the STOP condition on the I3C Bus after the complete data byte is transferred or received. The Driver clears the ABORT bit to allow operation on the Bus.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (RUNNING): The Host Controller is Running. ■ 0x1 (ABORTED): The Host Controller has Aborted a transfer. <p>Value After Reset: 0x0</p> <p>Exists: IC_DEVICE_ROLE<4</p> <p>Volatile: true</p>
28:9			Reserved Field: Yes
8	HOT_JOIN_CTRL	R/W	<p>Hot-Join ACK/NACK Control</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): ACK Hot-Join request ■ 0x1 (ENABLED): NACK and send broadcast CCC to disable Hot-Join <p>Value After Reset: IC_DFLT_HJ_CTRL</p> <p>Exists: IC_DEVICE_ROLE<4</p>

Table 5-9 Fields for Register: HC_CONTROL (Continued)

Bits	Name	Memory Access	Description
7	I2C_SLAVE_PRESENT	R/W	<p>I2C Slave Present on Bus.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (NOT_PRESENT): Legacy inclusive Bus mode disabled ■ 0x1 (PRESENT): Legacy inclusive (mix) Bus mode enabled <p>Value After Reset: IC_DFLT_I2C_SLAVE_PRESENT Exists: IC_DEVICE_ROLE<4</p>
6:1			Reserved Field: Yes
0	IBA_INCLUDE	R/W	<p>Include I3C Broadcast Address. This bit controls whether the I3C broadcast address (0x7E) is included for private transfers. If the I3C broadcast address is not included for private transfers, then In-band Interrupts (IBI) driven from Slaves might not win arbitration, potentially delaying acceptance of the IBIs.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (NOT_INCLUDED): Do not include I3C Broadcast Address for Private Transfers ■ 0x1 (INCLUDED): Include I3C Broadcast Address for Private Transfers <p>Value After Reset: IC_DFLT_IBA_INC Exists: IC_DEVICE_ROLE<4</p>

5.1.3 MASTER_DEVICE_ADDR

- **Name:** Master Device Address Register
- **Description:** Master Device Address Register
- **Size:** 32 bits
- **Offset:** 0x8
- **Exists:** (IC_DEVICE_ROLE<5)

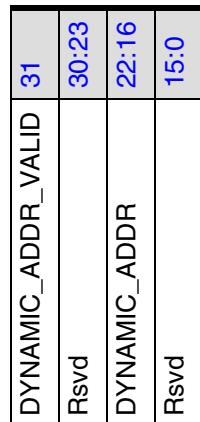


Table 5-10 Fields for Register: MASTER_DEVICE_ADDR

Bits	Name	Memory Access	Description
31	DYNAMIC_ADDR_VALID	R/W	<p>Device dynamic address valid</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (INVALID): Dynamic Address Field not valid ■ 0x1 (VALID): Dynamic Address Field valid <p>Value After Reset: IC_DFLT_DYNAMIC_ADDR_VALID</p> <p>Exists: Always</p> <p>Volatile: true</p>
30:23			Reserved Field: Yes
22:16	DYNAMIC_ADDR	R/W	<p>Device Dynamic Address</p> <p>Value After Reset: IC_DFLT_DYNAMIC_ADDR</p> <p>Exists: Always</p> <p>Volatile: true</p>
15:0			Reserved Field: Yes

5.1.4 HC CAPABILITIES

- **Name:** HC Capabilities Register
 - **Description:** Device Capabilities register identifies capabilities of the Master Host Controller hardware.
 - **Size:** 32 bits
 - **Offset:** 0xc
 - **Exists:** IC_DEVICE_ROLE<4

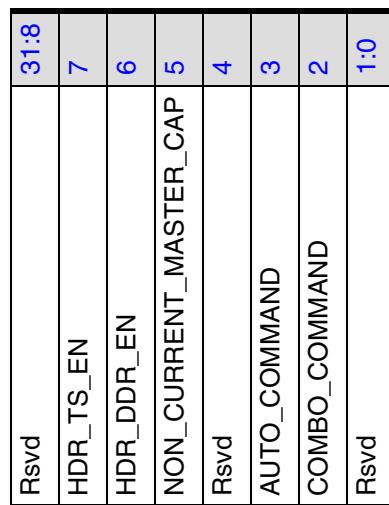


Table 5-11 Fields for Register: HC_CAPABILITIES

Bits	Name	Memory Access	Description
31:8			Reserved Field: Yes
7	HDR_TS_EN	R	<p>Defines whether the Host Controller supports HDR-TS transfers.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): HDR-TS not supported ■ 0x1 (ENABLED): HDR-TS supported <p>Value After Reset: IC_SPEED_HDR_TS</p> <p>Exists: Always</p>

Table 5-11 Fields for Register: HC_CAPABILITIES (Continued)

Bits	Name	Memory Access	Description
6	HDR_DDR_EN	R	<p>Defines whether the Host Controller supports HDR-DDR transfers.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): HDR-DDR not supported ■ 0x1 (ENABLED): HDR-DDR supported <p>Value After Reset: IC_SPEED_HDR_DDR</p> <p>Exists: Always</p>
5	NON_CURRENT_MASTER_CAP	R	<p>Defines whether the Host Controller supports handoff of the Current Master role to another Device on the I3C Bus.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): Not capable of acting as Non-Current Master. ■ 0x1 (ENABLED): Capable acting as Non-Current Master <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
4			Reserved Field: Yes
3	AUTO_COMMAND	R	<p>Defines whether Host Controller supports IBI Auto Command functionality.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): IBI Auto Command Transfers is not supported ■ 0x1 (ENABLED): IBI Auto Command Transfers is supported <p>Value After Reset: "(IC_HAS_DAT_EN==1 && IC_HAS_IBI_DATA==1) ? 1 : 0"</p> <p>Exists: Always</p>
2	COMBO_COMMAND	R	<p>Defines whether Host Controller supports Combo Command Transfers</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): Combo Command Transfers not supported ■ 0x1 (ENABLED): Combo Command Transfers supported <p>Value After Reset: 0x1</p> <p>Exists: Always</p>
1:0			Reserved Field: Yes

5.1.5 RESET_CONTROL

- **Name:** Reset Control Register
- **Description:** Reset Control register is used to reset specific functional areas of Host Controller, including buffer resets.
- **Size:** 32 bits
- **Offset:** 0x10
- **Exists:** (IC_DEVICE_ROLE<5)

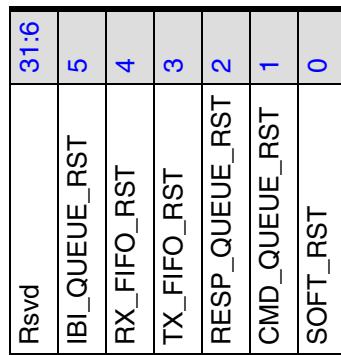


Table 5-12 Fields for Register: RESET_CONTROL

Bits	Name	Memory Access	Description
31:6			Reserved Field: Yes
5	IBI_QUEUE_RST	R/W	IBI Queue Reset Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4 Volatile: true
4	RX_FIFO_RST	R/W	Receive Queue Software reset Value After Reset: 0x0 Exists: Always Volatile: true
3	TX_FIFO_RST	R/W	Transmit Queue Software Reset Value After Reset: 0x0 Exists: Always Volatile: true

Table 5-12 Fields for Register: RESET_CONTROL (Continued)

Bits	Name	Memory Access	Description
2	RESP_QUEUE_RST	R/W	<p>Response Queue software reset</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p> <p>Volatile: true</p>
1	CMD_QUEUE_RST	R/W	<p>Command Queue Software reset</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p> <p>Volatile: true</p>
0	SOFT_RST	R/W	<p>Core software Reset</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p> <p>Volatile: true</p>

5.1.6 PRESENT_STATE

- **Name:** Present State Register
- **Description:** Present state debug register.
- **Size:** 32 bits
- **Offset:** 0x14
- **Exists:** IC_DEVICE_ROLE<4



Table 5-13 Fields for Register: PRESENT_STATE

Bits	Name	Memory Access	Description
31:3			Reserved Field: Yes
2	CURRENT_MASTER	R	<p>Current Master This bit indicates whether or not the Master is presently the Current Master on the I3C Bus. The Current Master is the Master Device on the I3C Bus that owns the SCL line.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (NOT_BUS_OWNER): The Master is not the Current Master, and must request and acquire Bus ownership before initiating any transfer. ■ 0x1 (BUS_OWNER): The Master is the Current Master, and as a result can initiate transfers. <p>Value After Reset: 0x0</p> <p>Exists: Always</p> <p>Volatile: true</p>
1:0			Reserved Field: Yes

5.1.7 INTR_STATUS

- **Name:** Interrupt Status Register
- **Description:** The Interrupt Status register reflects the status of outstanding interrupt(s). The status fields are either RW1C (write 1 to clear), or else are cleared based on queue operations.
- **Size:** 32 bits
- **Offset:** 0x20
- **Exists:** (IC_DEVICE_ROLE<5)

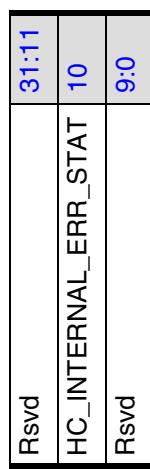


Table 5-14 Fields for Register: INTR_STATUS

Bits	Name	Memory Access	Description
31:11			Reserved Field: Yes
10	HC_INTERNAL_ERR_STAT	R/W1C	<p>Host Controller Internal Error reflects the interrupt status that is populated upon non-recoverable internal error of the Host Controller</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p> <p>Volatile: true</p>
9:0			Reserved Field: Yes

5.1.8 INTR_STATUS_ENABLE

- **Name:** Interrupt Status Enable Register
- **Description:** The Interrupt Status Enable register enables or disables reporting of outstanding interrupts.
- **Size:** 32 bits
- **Offset:** 0x24
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-15 Fields for Register: INTR_STATUS_ENABLE

Bits	Name	Memory Access	Description
31:11			Reserved Field: Yes
10	HC_INTERNAL_ERR_STAT_EN	R/W	Host Controller Internal Error Status Enable Value After Reset: 0x0 Exists: Always
9:0			Reserved Field: Yes

5.1.9 INTR_SIGNAL_ENABLE

- **Name:** Interrupt Signal Enable Register
- **Description:** The Interrupt Signal Enable register is enables or disables signalling of outstanding interrupts received by Host Controller.
- **Size:** 32 bits
- **Offset:** 0x28
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-16 Fields for Register: INTR_SIGNAL_ENABLE

Bits	Name	Memory Access	Description
31:11			Reserved Field: Yes
10	HC_INTERNAL_ERR_SIGNAL_EN	R/W	Host Controller Internal Error Signal Enable Value After Reset: 0x0 Exists: Always
9:0			Reserved Field: Yes

5.1.10 INTR_FORCE

- **Name:** Interrupt Force Enable Register
- **Description:** The Interrupt Force register is used to force a specific interrupt. It can be used for debugging purposes.
- **Size:** 32 bits
- **Offset:** 0x2c
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-17 Fields for Register: INTR_FORCE

Bits	Name	Memory Access	Description
31:11			Reserved Field: Yes
10	HC_INTERNAL_ERR_FORCE	W	Host Controller Internal Error Force Enable Value After Reset: 0x0 Exists: Always
9:0			Reserved Field: Yes

5.1.11 DAT_SECTION_OFFSET

- **Name:** Device Address Table Pointer
- **Description:** The Device Address Table Section Offset register holds the offset and size of the DAT table.
- **Size:** 32 bits
- **Offset:** 0x30
- **Exists:** IC_DEVICE_ROLE<4 && IC_HAS_HCI==1 && IC_HAS_DAT==1

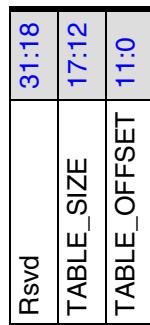


Table 5-18 Fields for Register: DAT_SECTION_OFFSET

Bits	Name	Memory Access	Description
31:18			Reserved Field: Yes
17:12	TABLE_SIZE	R	DAT Table Size Size of the DAT, in DWORDS. Value After Reset: IC_DEV_ADDR_TABLE_BUF_DEPTH Exists: Always
11:0	TABLE_OFFSET	R	DAT Table Offset Offset of the DAT relative to the BASE address of the current HCI. If the Host Controller does not support implementing the DAT in registers (thereby forcing the software to provide the Device Context in memory), then it should indicate this by setting this field to 12'h000. Value After Reset: IC_DFLT_DAT_SEC_OFFSET Exists: Always

5.1.12 DCT_SECTION_OFFSET

- **Name:** Device Characteristics Table Pointer
- **Description:** The Device Characteristics Table Section Offset register holds the offset and size of the DCT table.
- **Size:** 32 bits
- **Offset:** 0x34
- **Exists:** IC_DEVICE_ROLE<4&&IC_HAS_DAA==1 && IC_HAS_HCI==1 && IC_HAS_DAT==1

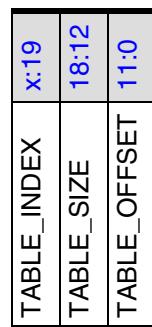


Table 5-19 Fields for Register: DCT_SECTION_OFFSET

Bits	Name	Memory Access	Description
x:19	TABLE_INDEX	R/W	<p>DCT Table Index Current index of the DCT, which is used as the starting index for the I3C ENTDAA CCC. Once the complete characteristics of device that won the arbitration are written to the DCT (during ENTDAA using Address Assignment Command) this index is incremented by 1. If needed, the software may override starting DCT index by setting this field.</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_HAS_DAA == 1) && (IC_RAM_DAA_ABW > 2)</p> <p>Range Variable[x]: IC_RAM_DAA_ABW + 16</p>
18:12	TABLE_SIZE	R	<p>DCT Table Size Size of the DCT, in DWORDs.</p> <p>Value After Reset: IC_DEV_CHAR_TABLE_BUF_DEPTH</p> <p>Exists: Always</p>

Table 5-19 Fields for Register: DCT_SECTION_OFFSET (Continued)

Bits	Name	Memory Access	Description
11:0	TABLE_OFFSET	R	<p>DCT Table Offset Offset of the DCT relative to the BASE address of the current HCI. If the Host Controller does not support implementing the DCT in registers, thereby forcing the Software to provide the Device Context, then it should indicate this by providing this field to 12'h000.</p> <p>Value After Reset: IC_DFLT_DCT_SEC_OFFSET</p> <p>Exists: Always</p>

5.1.13 RING_HEADERS_SECTION_OFFSET

- **Name:** Ring Headers Section Offset Register
- **Description:** The Ring Headers Section Offset register indicates the location of the Ring Headers Section of the register map.
- **Size:** 32 bits
- **Offset:** 0x38
- **Exists:** IC_HAS_HCI==1



Table 5-20 Fields for Register: RING_HEADERS_SECTION_OFFSET

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15:0	SECTION_OFFSET	R	Ring Headers Section Offset Offset of the Ring Headers registers section of the register map, relative to the BASE address of the current HCI. A value of 0 in this register indicates that the Ring Headers section is not implemented. Value After Reset: IC_DFLT_RING_HDR_SEC_OFFSET Exists: Always

5.1.14 PIO_SECTION_OFFSET

- **Name:** PIO Section Offset Register
- **Description:** The PIO Section Offset register indicates the location of the PIO Section of the register map.

- **Size:** 32 bits
- **Offset:** 0x3c
- **Exists:** IC_HAS_HCI==1

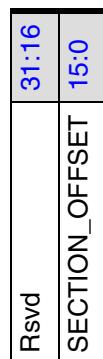


Table 5-21 Fields for Register: PIO_SECTION_OFFSET

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15:0	SECTION_OFFSET	R	<p>PIO Section Offset Offset of the PIO registers section of the register map, relative to the BASE address of the current HCI. A value of 0 in this field indicates the PIO section is not implemented.</p> <p>Value After Reset: IC_DFLT_PIO_SEC_OFFSET</p> <p>Exists: Always</p>

5.1.15 EXTCAPS_SECTION_OFFSET

- **Name:** Extended Capabilities Section Offset Register
- **Description:** The Extended Capabilities Section Offset register indicates the location of the Extended Capabilities section Of the register map.
- **Size:** 32 bits
- **Offset:** 0x40
- **Exists:** IC_HAS_HCI==1



Table 5-22 Fields for Register: EXTCAPS_SECTION_OFFSET

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15:0	SECTION_OFFSET	R	<p>Extended Capabilities Section Offset Offset of the Extended Capabilities registers section of the register map, relative to the BASE address of the current HCI. A value of 0 in this field indicates that the Extended Capabilities section is not implemented.</p> <p>Value After Reset: IC_DFLT_EXTCAPS_SEC_OFFSET Exists: Always</p>

5.1.16 IBI_NOTIFY_CTRL

- **Name:** IBI Queue Control Register
- **Description:** The IBI Notify Control register enables or disables event notifications for the IBI Queue/Ring.
- **Size:** 32 bits
- **Offset:** 0x58
- **Exists:** IC_DEVICE_ROLE<4

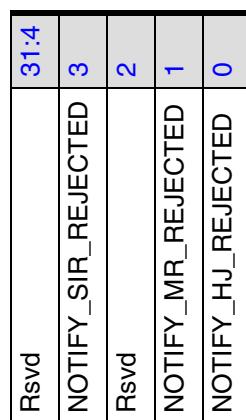


Table 5-23 Fields for Register: IBI_NOTIFY_CTRL

Bits	Name	Memory Access	Description
31:4			Reserved Field: Yes
3	NOTIFY_SIR_REJECTED	R/W	<p>Notify Rejected Slave Interrupt Request Control Enables or disables reporting rejection of individual Slave Interrupt Requests (SIR).</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): Do not pass rejected IBI Status to the IBI Queue/Rings, if the incoming SIR is NACKed and is auto-disabled based on SIR_REJECT field in relevant DAT entry. ■ 0x1 (ENABLED): Pass rejected IBI Status to the IBI Queue/Rings, if the incoming SIR is NACKed and is auto-disabled based on SIR_REJECT field in relevant DAT entry. <p>Value After Reset: IC_DFLT_SIR_REJECTED</p> <p>Exists: Always</p>
2			Reserved Field: Yes

Table 5-23 Fields for Register: IBI_NOTIFY_CTRL (Continued)

Bits	Name	Memory Access	Description
1	NOTIFY_MR_REJECTED	R/W	<p>Notify Rejected Master Request Control Enables or disables reporting rejection of individual Master Requests.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): Do not pass rejected IBI Status to IBI Queue/Ring, if the incoming Master Request is NACKed and is auto-disabled based on MR_REJECT field in relevant DAT entry. ■ 0x1 (ENABLED): Pass rejected IBI Status to the IBI Queue, if the incoming Master Request is NACKed and is auto-disabled based on MR_REJECT field in relevant DAT entry. <p>Value After Reset: IC_DFLT_MR_REJECTED</p> <p>Exists: Always</p>
0	NOTIFY_HJ_REJECTED	R/W	<p>Notify Rejected Hot-Join Control Enables or disables reporting rejection of individual Hot Join requests.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): Do not pass rejected IBI Status to IBI Queue, if the incoming Hot-Join request is NACKed and is auto-disabled based on field HOT_JOIN_CTRL of HC_CONTROL. ■ 0x1 (ENABLED): Pass rejected IBI Status to the IBI Queue, if the incoming Hot Join request is NACKed and is auto-disabled based on field HOT_JOIN_CTRL of HC_CONTROL. <p>Value After Reset: IC_DFLT_HJ_REJECTED</p> <p>Exists: Always</p>

5.1.17 DEV_CTX_BASE_LO

- **Name:** Device Context Address Low Register
- **Description:** The Device Context Address Low register indicates the location of the Device Context (DAT & DCT) when provided in memory.
- **Size:** 32 bits
- **Offset:** 0x60
- **Exists:** IC_DEVICE_ROLE<4



Table 5-24 Fields for Register: DEV_CTX_BASE_LO

Bits	Name	Memory Access	Description
31:0	BASE_LO	R	<p>Device Context Base Low Lower 32 bits of pointer to physical memory allocated for the Device Context. The Device Context is DWORD aligned, so the last two address bits are always 2'b00.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.1.18 DEV_CTX_BASE_HI

- **Name:** Device Context Address High Register
- **Description:** The Device Context Address High register indicates the location of the Device Context (DAT & DCT) when provided in memory.
- **Size:** 32 bits
- **Offset:** 0x64
- **Exists:** IC_DEVICE_ROLE<4



Table 5-25 Fields for Register: DEV_CTX_BASE_HI

Bits	Name	Memory Access	Description
31:0	BASE_HI	R	Device Context Base High Upper 32 bits of pointer to physical memory allocated for Device Context. Value After Reset: 0x0 Exists: Always

5.1.19 HW_IDENTIFICATION_HEADER

- **Name:** Hardware Identification header Register
- **Description:** Extended capability header register for Hardware identification registers. Every Extended Capability is introduced with Extended Capability Header that comprises single EXTCAP_HEADER register and a number of capability specific register.

- **Size:** 32 bits
- **Offset:** 0x100
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-26 Fields for Register: HW_IDENTIFICATION_HEADER

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:8	CAP_LEN	R	Length of capability structure. Value After Reset: IC_DFLT_HW_CAP_LEN Exists: Always
7:0	CAP_ID	R	Extended capability ID. Value After Reset: IC_DFLT_HW_CAP_ID Exists: Always

5.1.20 COMP_MANUFACTURER

- **Name:** DWC_mipi_i3c MIPI Manufacturer ID Register
- **Description:** This register reflects the MIPI Assigned Manufacturer ID.

- **Size:** 32 bits
- **Offset:** 0x104
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-27 Fields for Register: COMP_MANUFACTURER

Bits	Name	Memory Access	Description
31:0	MIPI_VENDOR_ID	R	<p>MIPI Assigned Manufacturer ID This field indicates the MIPI Assigned Manufacturer ID of DWC_mipi_i3c Component.</p> <p>Value After Reset: IC_SLV_MIPI_MFG_ID Exists: Always</p>

5.1.21 COMP_VERSION

- **Name:** DWC_mipi_i3c Version ID Register
- **Description:** This register reflects the current release number of DWC_mipi_i3c

- **Size:** 32 bits
- **Offset:** 0x108
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-28 Fields for Register: COMP_VERSION

Bits	Name	Memory Access	Description
31:0	I3C_VER_ID	R	<p>Current release number</p> <p>This field indicates the Synopsys DesignWare Cores DWC_mipi_i3c current release number that is read by an application.</p> <p>For example, release number "1.00a" is represented in ASCII as 0x313030. Lower 8 bits read from this register can be ignored by the application. An application reading this register along with the I3C_VER_TYPE register, gathers details of the current release.</p> <p>Value After Reset: IC_DFLT_VER_ID</p> <p>Exists: Always</p>

5.1.22 COMP_TYPE

- **Name:** DWC_mipi_i3c Version Type Register
- **Description:** This register reflects the current release type of DWC_mipi_i3c.

- **Size:** 32 bits
- **Offset:** 0x10c
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-29 Fields for Register: COMP_TYPE

Bits	Name	Memory Access	Description
31:0	I3C_VER_TYPE	R	<p>Current release type</p> <p>This field indicates the Synopsys DesignWare Cores DWC_mipi_i3c current release type that is read by an application.</p> <p>For example, release type "ga" is represented in ASCII as 0x6761 and "ea" is represented as 0x6561. Lower 16 bits read from this register can be ignored by the application if release type is "ga". If release type is "ea" the lower 16 bits represents the "ea" release version.</p> <p>An application reading this register along with the I3C_VER_ID register, gathers details of the current release.</p> <p>Value After Reset: IC_DFLT_VER_TYPE</p> <p>Exists: Always</p>

5.1.23 BUS_TIMING_HEADER

- **Name:** Bus Timing Header Register
- **Description:** Extended capability header register for Bus Timing registers.

Every Extended Capability is introduced with Extended Capability Header that comprises single EXTCAP_HEADER register and a number of capability specific register.

- **Size:** 32 bits
- **Offset:** 0x110
- **Exists:** (IC_DEVICE_ROLE<5)

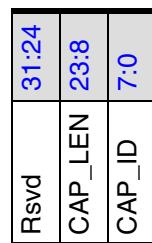


Table 5-30 Fields for Register: BUS_TIMING_HEADER

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:8	CAP_LEN	R	Length of capability structure. Value After Reset: IC_DFLT_BUS_TIMING_CAP_LEN Exists: Always
7:0	CAP_ID	R	Extended capability ID. Value After Reset: IC_DFLT_BUS_TIMING_CAP_ID Exists: Always

5.1.24 SCL_I3C_OD_TIMING

- **Name:** SCL I3C Open Drain Timing Register
- **Description:** SCL I3C Open Drain Timing Register

This register sets the SCL clock high period and low period count for I3C Open Drain transfers. The count value takes the number of core_clks to maintain the I/O SCL High/Low Period timing.

- **Size:** 32 bits
- **Offset:** 0x114
- **Exists:** IC_DEVICE_ROLE<4

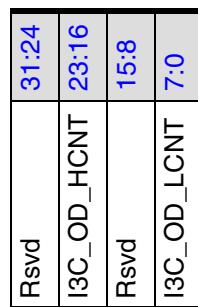


Table 5-31 Fields for Register: SCL_I3C_OD_TIMING

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:16	I3C_OD_HCNT	R/W	<p>I3C Open Drain High Count. SCL open-drain High count (I3C) for I3C transfers targeted to I3C devices.</p> <p>Value After Reset: IC_DFLT_I3C_OD_HCNT Exists: Always</p>
15:8			Reserved Field: Yes
7:0	I3C_OD_LCNT	R/W	<p>I3C Open Drain Low Count. SCL Open-drain low count for I3C transfers targeted to I3C devices.</p> <p>Value After Reset: IC_DFLT_I3C_OD_LCNT Exists: Always</p>

5.1.25 SCL_I3C_PP_TIMING

- **Name:** SCL I3C Push Pull Timing Register
- **Description:** SCL I3C Push Pull Timing Register

This register sets the SCL clock high period and low period count for I3C Push Pull transfers. The count value takes the number of core_clks to maintain the I/O SCL High/Low Period timing.

- **Size:** 32 bits
- **Offset:** 0x118
- **Exists:** IC_DEVICE_ROLE<4

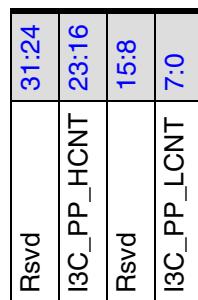


Table 5-32 Fields for Register: SCL_I3C_PP_TIMING

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:16	I3C_PP_HCNT	R/W	I3C Push Pull High Count. SCL push-pull High count for I3C transfers targeted to I3C devices. Value After Reset: IC_DFLT_I3C_PP_HCNT Exists: Always
15:8			Reserved Field: Yes
7:0	I3C_PP_LCNT	R/W	I3C Push Pull Low Count. SCL Push-pull low count for I3C transfers targeted to I3C devices. Value After Reset: IC_DFLT_I3C_PP_LCNT Exists: Always

5.1.26 SCL_I2C_FM_TIMING

- **Name:** SCL I2C Fast Mode Timing Register
- **Description:** SCL I2C Fast Mode Timing Register

This register sets the SCL clock high period and low period count for I2C Fast Mode transfers. The count value takes the number of core_clks to maintain the I/O SCL Low/High period timing.

- **Size:** 32 bits
- **Offset:** 0x11c
- **Exists:** IC_DEVICE_ROLE<4

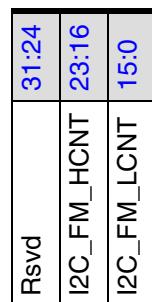


Table 5-33 Fields for Register: SCL_I2C_FM_TIMING

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:16	I2C_FM_HCNT	R/W	<p>I2C Fast Mode High Count The SCL open-drain high count timing for I2C fast mode transfers.</p> <p>Value After Reset: IC_DFLT_I2C_FM_HCNT Exists: Always</p>
15:0	I2C_FM_LCNT	R/W	<p>I2C Fast Mode Low Count The SCL open-drain low count timing for I2C fast mode transfers.</p> <p>Value After Reset: IC_DFLT_I2C_FM_LCNT Exists: Always</p>

5.1.27 SCL_I2C_FMP_TIMING

- **Name:** SCL I2C Fast Mode Plus Timing Register
- **Description:** SCL I2C Fast Mode Plus Timing Register

This register sets the SCL clock high period and low period count for I2C Fast Mode Plus transfers. The count value takes the number of core_clks to maintain the I/O SCL Low/High period timing.

- **Size:** 32 bits
- **Offset:** 0x120
- **Exists:** IC_DEVICE_ROLE<4



Table 5-34 Fields for Register: SCL_I2C_FMP_TIMING

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:16	I2C_FMP_HCNT	R/W	I2C Fast Mode Plus High Count The SCL open-drain high count timing for I2C fast mode plus transfers. Value After Reset: IC_DFLT_I2C_FMP_HCNT Exists: Always
15:8			Reserved Field: Yes
7:0	I2C_FMP_LCNT	R/W	I2C Fast Mode Plus Low Count The SCL open-drain low count timing for I2C fast mode plus transfers. Value After Reset: IC_DFLT_I2C_FMP_LCNT Exists: Always

5.1.28 SCL_I2C_SS_TIMING

- **Name:** SCL I2C Standard Speed Timing Register
- **Description:** SCL I2C Standard Speed Timing Register

This register sets the SCL clock high period and low period count for I2C Fast Mode transfers. The count value takes the number of core_clks to maintain the I/O SCL Low/High period timing.

- **Size:** 32 bits
- **Offset:** 0x124
- **Exists:** IC_DEVICE_ROLE<4



Table 5-35 Fields for Register: SCL_I2C_SS_TIMING

Bits	Name	Memory Access	Description
31:16	I2C_SS_HCNT	R/W	<p>I2C Standard Speed High Count The SCL open-drain high count timing for I2C standard speed transfers.</p> <p>Value After Reset: IC_DFLT_I2C_SS_HCNT Exists: Always</p>
15:0	I2C_SS_LCNT	R/W	<p>I2C Standard Speed Low Count The SCL open-drain low count timing for I2C standard speed transfers.</p> <p>Value After Reset: IC_DFLT_I2C_SS_LCNT Exists: Always</p>

5.1.29 SCL_EXT_LCNT_TIMING

- **Name:** SCL Extended Low Count Timing Register
- **Description:** SCL Extended Low Count Timing Register.

This register sets the extended low periods for the I3C transfers to allow the low data rates of the Slave devices as specified in GETMXDS CCC. The Speed field of Transfer command (COMMAND_QUEUE_PORT_TRANSFER_COMMAND) decides the selection of extended low period to achieve the lower data rate for the transfers to Slave devices.

- SDR1: Uses I3C_EXT_LCNT_1 field for the data transfer.
- SDR2: Uses I3C_EXT_LCNT_2 field for the data transfer.
- SDR3: Uses I3C_EXT_LCNT_3 field for the data transfer.
- SDR4: Uses I3C_EXT_LCNT_4 field for the data transfer.

- **Size:** 32 bits
- **Offset:** 0x128
- **Exists:** IC_DEVICE_ROLE<4

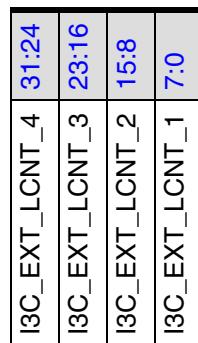


Table 5-36 Fields for Register: SCL_EXT_LCNT_TIMING

Bits	Name	Memory Access	Description
31:24	I3C_EXT_LCNT_4	R/W	<p>I3C Extended Low Count Register 4 SDR4 uses this register field for data transfer.</p> <p>Value After Reset: IC_DFLT_EXT_LCNT_4 Exists: Always</p>

Table 5-36 Fields for Register: SCL_EXT_LCNT_TIMING (Continued)

Bits	Name	Memory Access	Description
23:16	I3C_EXT_LCNT_3	R/W	<p>I3C Extended Low Count Register 3 SDR3 uses this register field for data transfer.</p> <p>Value After Reset: IC_DFLT_EXT_LCNT_3 Exists: Always</p>
15:8	I3C_EXT_LCNT_2	R/W	<p>I3C Extended Low Count Register 2 SDR2 uses this register field for data transfer.</p> <p>Value After Reset: IC_DFLT_EXT_LCNT_2 Exists: Always</p>
7:0	I3C_EXT_LCNT_1	R/W	<p>I3C Extended Low Count Register 1 SDR1 uses this register field for data transfer.</p> <p>Value After Reset: IC_DFLT_EXT_LCNT_1 Exists: Always</p>

5.1.30 SCL_EXT_TERMN_LCNT_TIMING

- **Name:** SCL Termination Bit Low count Timing Register
- **Description:** SCL Termination Bit Low Count Timing Register

This register is used to extend the SCL Low period for Read Termination Bit.

- **Size:** 32 bits
- **Offset:** 0x12c
- **Exists:** IC_DEVICE_ROLE<4

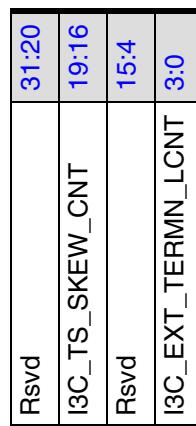


Table 5-37 Fields for Register: SCL_EXT_TERMN_LCNT_TIMING

Bits	Name	Memory Access	Description
31:20			Reserved Field: Yes
19:16	I3C_TS_SKEW_CNT	R/W	I3C HDR Ternary Skew Count. Ternary Skew Count in terms of core_clks which is used for HDR Ternary Bus Turn-Around Detection in Master Mode. Value After Reset: IC_DFLT_TS_SKEW_LCNT Exists: IC_DEVICE_ROLE<4 && IC_SPEED_HDR_TS==1
15:4			Reserved Field: Yes

Table 5-37 Fields for Register: SCL_EXT_TERMN_LCNT_TIMING (Continued)

Bits	Name	Memory Access	Description
3:0	I3C_EXT_TERMN_LCNT	R/W	<p>I3C Read Termination Bit Low count. Extended I3C Read Termination Bit low count for I3C Read transfers. Effective Termination-Bit Low Period is derived based on the SDR speed as shown below</p> <ul style="list-style-type: none"> ■ SDR0 speed: I3C_PP_LCNT + I3C_EXT_TERMN_LCNT ■ SDR1 speed: I3C_EXT_LCNT_1 + I3C_EXT_TERMN_LCNT ■ SDR2 speed: I3C_EXT_LCNT_2 + I3C_EXT_TERMN_LCNT ■ SDR3 speed: I3C_EXT_LCNT_3 + I3C_EXT_TERMN_LCNT ■ SDR4 speed: I3C_EXT_LCNT_4 + I3C_EXT_TERMN_LCNT <p>Value After Reset: IC_DFLT_TERMN_LCNT Exists: Always</p>

5.1.31 SDA_HOLD_SWITCH_DLY_TIMING

- **Name:** SDA Hold and Mode Switch Delay Timing Register
- **Description:** SDA Hold and Mode Switch Delay Timing Register

The Bits [2:0] of this register are used to shift the sda_out with respect to sda_oe while switching transfer from Open Drain timing to Push Pull timing. The bits [10:8] of this register are used to shift the sda_oe with respect to sda_out while switching transfer from Push pull timing to Open Drain timing. The bits [18:16] of this register are used to control the hold time of SDA during transmit mode in SDR & DDR transfers.

- **Size:** 32 bits
- **Offset:** 0x130
- **Exists:** IC_DEVICE_ROLE<4

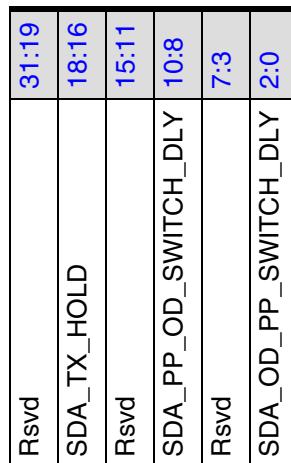


Table 5-38 Fields for Register: SDA_HOLD_SWITCH_DLY_TIMING

Bits	Name	Memory Access	Description
31:19			Reserved Field: Yes
18:16	SDA_TX_HOLD	R/W	<p>This field controls the hold time (in term of the core clock period) of the transmit data (SDA) with respect to the SCL edge in SD FM FM+ SDR and DDR speed mode of operations. This field is not applicable for the ternary speed modes. The valid values are 1 to 7. Others are Reserved.</p> <p>Value After Reset: IC_DFLT_SDA_TX_HOLD Exists: Always</p>

Table 5-38 Fields for Register: SDA_HOLD_SWITCH_DLY_TIMING (Continued)

Bits	Name	Memory Access	Description
15:11			Reserved Field: Yes
10:8	SDA_PP_OD_SWITCH_DLY	R/W	<p>This field is used to delay the sda_oe with respect to sda_out (in terms of the core clock period) while switching the transfer from PP1 (Push-Pull Mode SDA=1) to OD1 (Open-Drain SDA=1). The valid value can range from 0 to 4. Others are Reserved.</p> <p>Value After Reset: IC_DFLT_PP_OD_SWITCH_DLY Exists: Always</p>
7:3			Reserved Field: Yes
2:0	SDA_OD_PP_SWITCH_DLY	R/W	<p>This field is used to delay the sda_out with respect to sda_oe while switching the transfer from OD1 (Open-Drain Mode SDA=1) to PP1 (Push-Pull Mode SDA=1). The valid value can range from 0 to 4. Others are reserved.</p> <p>Value After Reset: IC_DFLT_OD_PP_SWITCH_DLY Exists: Always</p>

5.1.32 BUS_FREE_TIMING

- **Name:** Bus Free Timing Register
- **Description:** Bus Free Timing Register

This register sets the Bus free time for initiating the transfer in master mode or generating IBI in non-current master mode.

- **Size:** 32 bits
- **Offset:** 0x134
- **Exists:** IC_DEVICE_ROLE<4

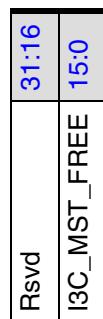


Table 5-39 Fields for Register: BUS_FREE_TIMING

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15:0	I3C_MST_FREE	R/W	<p>I3C Master Free Count Value. In Pure Bus System, this field represents tCAS parameter. In Mixed Bus system, this field is expected to be programmed to tLOW of I2C Timing.</p> <p>Value After Reset: IC_DFLT_BUS_FREE_CNT Exists: Always</p>

5.1.33 SCL_LOW_MST_EXT_TIMEOUT

- **Name:** SCL_LOW_MST_EXT_TIMEOUT
- **Description:** The SCL Low Master Extended Timeout register is used to define the duration of the SCL Low Bus Reset Pattern.

- **Size:** 32 bits
- **Offset:** 0x13c
- **Exists:** IC_HAS_DEVICE_RESET_SUPPORT==1



Table 5-40 Fields for Register: SCL_LOW_MST_EXT_TIMEOUT

Bits	Name	Memory Access	Description
x:0	SCL_LOW_MST_TIMEOUT_CO UNT	R/W	<p>This count defines the number of core clock periods to count for generation of the SCL Low Bus Reset Pattern.</p> <p>Value After Reset: IC_DFLT_SCL_LOW_TIMEOUT_COUNT</p> <p>Exists: Always</p> <p>Range Variable[x]: IC_SCL_LOW_TIMEOUT_COUNT_WIDTH - 1</p>

5.1.34 DS_EXTCAP_HEADER

- **Name:** Debug specific header Register
- **Description:** Extended capability header register for Debug Specific registers.

Every Extended Capability is introduced with Extended Capability Header that comprises single EXTCAP_HEADER register and a number of capability specific register.

- **Size:** 32 bits
- **Offset:** 0x140
- **Exists:** (IC_DEVICE_ROLE<5)

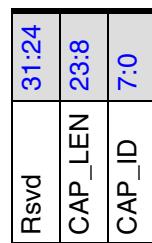


Table 5-41 Fields for Register: DS_EXTCAP_HEADER

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:8	CAP_LEN	R	Length of capability structure. Value After Reset: IC_DFLT_DEBUG_CAP_LEN Exists: Always
7:0	CAP_ID	R	Extended capability ID. Value After Reset: IC_DFLT_DEBUG_CAP_ID Exists: Always

5.1.35 QUEUE_STATUS_LEVEL

- **Name:** Queue Status Level Register
- **Description:** Queue Status Level Register.

- **Size:** 32 bits
- **Offset:** 0x144
- **Exists:** (IC_DEVICE_ROLE<5)

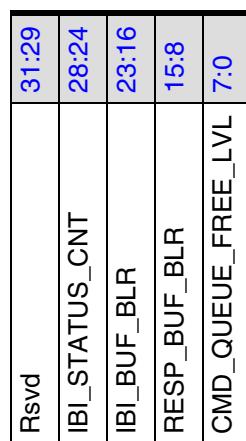


Table 5-42 Fields for Register: QUEUE_STATUS_LEVEL

Bits	Name	Memory Access	Description
31:29			Reserved Field: Yes
28:24	IBI_STATUS_CNT	R	IBI Buffer Status Count. Contains the number of IBI status entries in the IBI Buffer. Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4
23:16	IBI_BUF_BLR	R	IBI Buffer Level Value. Contains the number of buffer entries in the IBI Buffer. Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4

Table 5-42 Fields for Register: QUEUE_STATUS_LEVEL (Continued)

Bits	Name	Memory Access	Description
15:8	RESP_BUF_BLR	R	<p>Response Buffer Level Value. Contains the number of buffer entries in the response Buffer.</p> <p>Value After Reset: 0x0 Exists: Always</p>
7:0	CMD_QUEUE_FREE_LVL	R	<p>Command Queue Empty Locations. Contains the number of free buffer entries in the command Buffer.</p> <p>Value After Reset: IC_CMD_BUF_DEPTH Exists: Always</p>

5.1.36 DATA_BUFFER_STATUS_LEVEL

- **Name:** Data Buffer Status Level Register
- **Description:** Data Buffer Status Level Register

- **Size:** 32 bits
- **Offset:** 0x148
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-43 Fields for Register: DATA_BUFFER_STATUS_LEVEL

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15:8	RX_BUF_LVL	R	<p>Receive Buffer Level Value. Contains the number of valid data entries in the receive Buffer.</p> <p>Value After Reset: 0x0 Exists: Always</p>
7:0	TX_BUF_FREE_LVL	R	<p>Transmit Buffer Empty Level Value. Contains the number of empty locations in the transmit Buffer.</p> <p>Value After Reset: IC_TX_BUF_DEPTH Exists: Always</p>

5.1.37 PRESENT_STATE_DEBUG

- **Name:** Present State Register
- **Description:** Present State debug register is used to get status of Host Controller. The present state of the Host Controller is divided into mandatory part (this register) and optional part for debug purposes (PRESENT_STATE_DEBUG), part of Debug Capability registers in Extended Capabilities list. The fields should not be repeated between both registers.
- **Size:** 32 bits
- **Offset:** 0x14c
- **Exists:** IC_DEVICE_ROLE<4

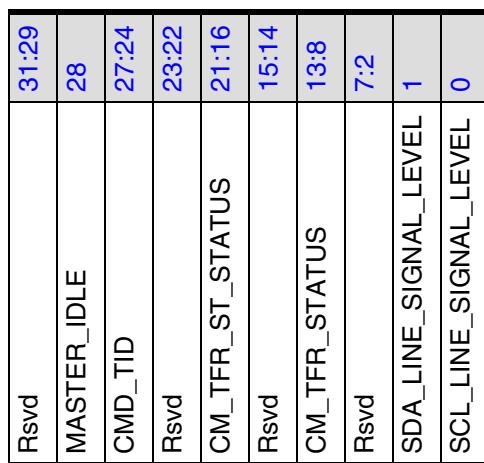


Table 5-44 Fields for Register: PRESENT_STATE_DEBUG

Bits	Name	Memory Access	Description
31:29			Reserved Field: Yes
28	MASTER_IDLE	R	<p>This field reflects whether the Master Controller is in Idle state or not. This bit is set when all the Queues(Command , Response, IBI) and Buffers(Transmit and Receive) are empty along with the Master State machine is in Idle state.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (MST_NOT_IDLE): Master Controller is not in IDLE State ■ 0x1 (MST_IDLE): Master Controller is in IDLE State. <p>Value After Reset: 0x1</p> <p>Exists: Always</p> <p>Volatile: true</p>

Table 5-44 Fields for Register: PRESENT_STATE_DEBUG (Continued)

Bits	Name	Memory Access	Description
27:24	CMD_TID	R	<p>This field reflects the Transaction-ID of the current executing command. The Transaction ID is optional, software defined tag for every command (opaque for Host Controller) Specifically it is useful for detection of currently executed command while scheduling transfers in PIO mode.</p> <p>Value After Reset: 0x0 Exists: Always</p>
23:22			Reserved Field: Yes

Table 5-44 Fields for Register: PRESENT_STATE_DEBUG (Continued)

Bits	Name	Memory Access	Description
21:16	CM_TFR_ST_STATUS	R	<p>Current Master Transfer State Status. Indicates the state of current transfer currently executing by the Host controller.</p> <ul style="list-style-type: none"> ■ 6'h0: IDLE (Controller is Idle state, waiting for commands from application or Slave initiated In-band Interrupt) ■ 6'h1: START Generation State. ■ 6'h2: RESTART Generation State. ■ 6'h3: STOP Generation State. ■ 6'h4: START Hold Generation for the Slave Initiated START State. ■ 6'h5: Broadcast Write Address Header(7'h7E,W) Generation State. ■ 6'h6: Broadcast Read Address Header(7'h7E,R) Generation State. ■ 6'h7: Dynamic Address Assignment State. ■ 6'h8: Slave Address Generation State. ■ 6'hB: CCC Byte Generation State. ■ 6'hC: HDR Command Generation State. ■ 6'hD: Write Data Transfer State. ■ 6'hE: Read Data Transfer State. ■ 6'hF: In-Band Interrupt(SIR) Address Read Data State. ■ 6'h10: In-Band Interrupt Auto-Disable State ■ 6'h11: HDR-DDR CRC Data Generation/Receive State. ■ 6'h12: Clock Extension State. ■ 6'h13: Halt State. ■ 6'h14: In-Band Interrupt(SIR) Read Data State. <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (IDLE) ■ 0x1 (START): START Generation State ■ 0x2 (RESTART): RESTART Generation State ■ 0x3 (STOP): STOP Generation State ■ 0x4 (START_HOLD): START Hold Generation for the Slave Initiated START State ■ 0x5 (BCAST_WRITE): Broadcast Write Address Header(7'h7E,W) Generation State

Table 5-44 Fields for Register: PRESENT_STATE_DEBUG (Continued)

Bits	Name	Memory Access	Description
			<ul style="list-style-type: none"> ■ 0x6 (BCAST_READ): Broadcast Read Address Header(7'h7E,R) Generation State ■ 0x7 (DAA): Dynamic Address Assignment State ■ 0x8 (SAG): Slave Address Generation State ■ 0xb (CCC): CCC Byte Generation State ■ 0xc (HDR): HDR Command Generation State ■ 0xd (WR): Write Data Transfer State ■ 0xe (RD): Read Data Transfer State ■ 0xf (SIR_READ): In-Band Interrupt(SIR) Read Data State ■ 0x10 (SIR_AUTO_DISABLE): In-Band Interrupt(SIR) Auto-Disable State ■ 0x11 (HDR_DDR_CRC): HDR-DDR CRC Data Generation/Receive State ■ 0x12 (CLOCK_EXT): Clock Extension State ■ 0x13 (HALT): Halt State ■ 0x14 (SIR_READ_DATA_STATE): In-Band Interrupt(SIR) Read Data State. <p>Value After Reset: 0x0 Exists: Always</p>
15:14			Reserved Field: Yes

Table 5-44 Fields for Register: PRESENT_STATE_DEBUG (Continued)

Bits	Name	Memory Access	Description
13:8	CM_TFR_STATUS	R	<p>Current Master Transfer Type Status. Indicates the type of transfer currently executing by the DWC_mipi_i3c controller.</p> <ul style="list-style-type: none"> ■ 6'h0: IDLE (Controller is in Idle state, waiting for commands from application or Slave initiated In-band Interrupt) ■ 6'h1: Broadcast CCC Write Transfer. ■ 6'h2: Directed CCC Write Transfer. ■ 6'h3: Directed CCC Read Transfer. ■ 6'h4: ENTDAA Address Assignment Transfer. ■ 6'h5: SETDASA Address Assignment Transfer. ■ 6'h6: Private I3C SDR Write Transfer. ■ 6'h7: Private I3C SDR Read Transfer. ■ 6'h8: Private I2C SDR Write Transfer. ■ 6'h9: Private I2C SDR Read Transfer. ■ 6'hA: Private HDR Ternary Symbol(TS) Write Transfer. ■ 6'hB: Private HDR Ternary Symbol(TS) Read Transfer. ■ 6'hC: Private HDR Double-Data Rate(DDR) Write Transfer. ■ 6'hD: Private HDR Double-Data Rate(DDR) Read Transfer. ■ 6'hE: Servicing In-Band Interrupt Transfer. ■ 6'hF: Halt state (Controller is in Halt State, waiting for the application to resume through DEVICE_CTRL Register) <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (IDLE) ■ 0x1 (BCAST_WRITE): Broadcast CCC Write Transfer ■ 0x2 (SLAVE_WRITE): Directed CCC Write Transfer ■ 0x3 (SLAVE_READ): Directed CCC Read Transfer ■ 0x4 (ENTDAA): ENTDAA Address Assignment Transfer ■ 0x5 (SETDASA): SETDASA Address Assignment Transfer ■ 0x6 (I3C_SDR_WRITE): Private I3C SDR Write Transfer ■ 0x7 (I3C_SDR_READ): Private I3C SDR Read Transfer

Table 5-44 Fields for Register: PRESENT_STATE_DEBUG (Continued)

Bits	Name	Memory Access	Description
			<ul style="list-style-type: none"> ■ 0x8 (I2C_SDR_WRITE): Private I2C SDR Write Transfer ■ 0x9 (I2C_SDR_READ): Private I2C SDR Read Transfer ■ 0xa (HDR_TS_WRITE): Private HDR Ternary Symbol(TS) Write Transfer ■ 0xb (HDR_TS_READ): Private HDR Ternary Symbol(TS) Read Transfer ■ 0xc (HDR_DDR_WRITE): Private HDR Double-Data Rate(DDR) Write Transfer ■ 0xd (HDR_DDR_READ): Private HDR Double-Data Rate(DDR) Read Transfer ■ 0xe (IBI): Servicing In-Band Interrupt Transfer ■ 0xf (HALTED): Controller is in Halt State, waiting for the application to resume through DEVICE_CTRL Register <p>Value After Reset: 0x0 Exists: Always</p>
7:2			Reserved Field: Yes
1	SDA_LINE_SIGNAL_LEVEL	R	<p>This bit is used to check the SDA line level to recover from errors and for debugging. This bit reflects the value of synchronized sda_in_a signal.</p> <p>Value After Reset: 0x1 Exists: Always Volatile: true</p>
0	SCL_LINE_SIGNAL_LEVEL	R	<p>This bit is used to check the SCL line level to recover from errors and for debugging. This bit reflects the value of synchronized scl_in_a signal.</p> <p>Value After Reset: 0x1 Exists: Always Volatile: true</p>

5.1.38 MASTER_EXT_HEADER

- **Name:** Master Extended specific header Register
- **Description:** Extended capability header register for Master Extended configuration Specific registers.

Every Extended Capability is introduced with Extended Capability Header that comprises single EXTCAP_HEADER register and a number of capability specific register.

- **Size:** 32 bits
- **Offset:** 0x154
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-45 Fields for Register: MASTER_EXT_HEADER

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:8	CAP_LEN	R	Length of capability structure. Value After Reset: IC_DFLT_MASTER_EXT_CAP_LEN Exists: Always
7:0	CAP_ID	R	Extended capability ID. Value After Reset: IC_DFLT_MASTER_EXT_CAP_ID Exists: Always

5.1.39 MASTER_CONFIG

- **Name:** Master Config Register
- **Description:** Master Config register reflects Master configuration parameters.

- **Size:** 32 bits
- **Offset:** 0x158
- **Exists:** IC_DEVICE_ROLE<4

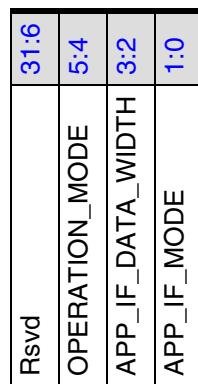


Table 5-46 Fields for Register: MASTER_CONFIG

Bits	Name	Memory Access	Description
31:6			Reserved Field: Yes
5:4	OPERATION_MODE	R	<p>Host Controller modes supported.</p> <ul style="list-style-type: none"> ■ 1: Master only ■ 2: Slave only ■ 3: Master or Slave <p>Value After Reset: IC_DEVICE_OPERATION_MODE Exists: Always</p>

Table 5-46 Fields for Register: MASTER_CONFIG (Continued)

Bits	Name	Memory Access	Description
3:2	APP_IF_DATA_WIDTH	R	<p>Host Controller Interface Data Width field is used to reflect data width configurable parameter.</p> <ul style="list-style-type: none"> ■ 0: 8bits ■ 1: 16bits ■ 2: 32bits ■ 3: 64bits <p>Value After Reset: 0x2 Exists: Always</p>
1:0	APP_IF_MODE	R	<p>Host Controller Interface Mode field is used to reflect Host Controller Mode configurable parameter.</p> <p>Value After Reset: IC_SLVIF_MODE Exists: Always</p>

5.1.40 COMMAND_QUEUE_PORT

- **Name:** COMMAND_QUEUE_PORT
- **Description:** COMMAND_QUEUE_PORT. Command Descriptor structure is used on Command Ring to schedule the command to devices on I3C bus. The Command Descriptor structure (64 bits) is used in two primary cases:
 - In PIO mode, the Command Descriptor is put to Command Queue through writes to Command Queue Port
 - In DMA mode, the Command Descriptor is put, as a part of Transfer Descriptor, on Command Ring

- **Size:** 32 bits
- **Offset:** 0x300
- **Exists:** IC_DEVICE_ROLE<4



Table 5-47 Fields for Register: COMMAND_QUEUE_PORT

Bits	Name	Memory Access	Description
31:0	COMMAND	W	32 bit command Value After Reset: 0x0 Exists: Always

5.1.41 RESPONSE_QUEUE_PORT

- **Name:** Response Queue Port Register
- **Description:** Response Queue Port Register

The Response Descriptor structure is used in two primary cases:

- In PIO mode, the Response Descriptor is read from Response Queue through reads from Response Queue Port.
- In DMA mode, the Response Descriptor is read from Response Ring.

- **Size:** 32 bits
- **Offset:** 0x304
- **Exists:** (IC_DEVICE_ROLE<5)

ERR_STATUS	31:28
TID	27:24
Rsvd	23:16
DATA_LENGTH	15:0

Table 5-48 Fields for Register: RESPONSE_QUEUE_PORT

Bits	Name	Memory Access	Description
31:28	ERR_STATUS	R	<p>Error Status defines the Error Type of the processed command</p> <ul style="list-style-type: none"> ■ 0x0: SUCCESS - Transfer successful, no error. ■ 0x1: CRC - CRC Error ■ 0x2: PARITY - Parity Error ■ 0x3: FRAME - Frame Error ■ 0x4: ADDR_HEADER - Address Header Error ■ 0x5: NACK - Address Nacked or Dynamic Address Assignment Nacked ■ 0x6: OVL - Receive Overflow or Transfer Underflow Error ■ 0x7: Reserved ■ 0x8: ABORTED ■ 0x9: Reserved ■ 0xA-0xF: Reserved <p>Value After Reset: 0x0 Exists: Always</p>
27:24	TID	R	<p>Transaction ID</p> <p>This field is used as the identification tag for the command. This value matches one of commands sent on the bus. You can represent 8 different ID's for the Command (4'b0000 to 4'b0111). The TID is used to link response with respective command. A command and its response has the same TID. The TID's from 4'b1000 to 4'b1110 are reserved. The TID 4'b1111 is used to indicate that CCCT field represents a CCC byte (Applicable only in Non-current Master mode).</p> <p>Value After Reset: 0x0 Exists: Always</p>
23:16			Reserved Field: Yes

Table 5-48 Fields for Register: RESPONSE_QUEUE_PORT (Continued)

Bits	Name	Memory Access	Description
15:0	DATA_LENGTH	R	<p>Data Length or Device Count</p> <ul style="list-style-type: none"> ■ For Write transfers, this field represents the remaining data length of the transfer. ■ For Read Transfers, this field represents the received data length in bytes. ■ For Address Assignment command, this field represents the remaining device count. <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4</p>

5.1.42 RX_DATA_PORT

- **Name:** Receive Data Port Register
- **Description:** Receive Data Port Register

This register when read from, reads data from the RX Buffer. This has the same offset as TX_DATA_PORT to provide a single bi-directional data port for transmitting or receiving the data from the DWC_mipi_i3c. Reset value of this register is not defined as it is mapped to External RAM.

- **Size:** 32 bits
- **Offset:** 0x308
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-49 Fields for Register: RX_DATA_PORT

Bits	Name	Memory Access	Description
31:0	RX_DATA_PORT	R	<p>Receive Data Port</p> <p>The Receive data port is mapped to the Rx-Data Buffer.</p> <p>The Receive data is always packed in 4-byte aligned and stored in the Rx-Data Buffer. If the command length is not aligned to the 4-bytes, then there are unused bytes in the end location of the transfer bytes.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.1.43 TX_DATA_PORT

- **Name:** Transmit Data Port Register
- **Description:** Transmit Data Port Register

This register when written into, writes data to the TX Buffer. This has the same offset as RX_DATA_PORT to provide a single bi-directional data port for transmitting or receiving the data from the DWC_mipi_i3c.

- **Size:** 32 bits
- **Offset:** 0x308
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-50 Fields for Register: TX_DATA_PORT

Bits	Name	Memory Access	Description
31:0	TX_DATA_PORT	W	<p>Transmit Data Port</p> <p>The Transmit Data port is mapped to the Tx-Data Buffer.</p> <p>The transmit data should always be packed in 4-byte aligned and written to the Transmit data port register. If the Command length is not aligned to the 4-bytes, then there are unused bytes in the end location of the transfer bytes.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.1.44 IBI_PORT

- **Name:** IBI Port Register
- **Description:** IBI Port Register is used to
 1. read IBI queue status descriptor
 2. read IBI queue data (raw/opaque data)

- **Size:** 32 bits
- **Offset:** 0x30c
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-51 Fields for Register: IBI_PORT

Bits	Name	Memory Access	Description
31:0	IBI_DATA	R	<p>Data read from IBI Data Buffer</p> <p>This register is mapped to IBI ring data buffer. The IBI data is always packed in 4-byte aligned and put to the IBI ring. If the incoming data is not aligned to the 4-bytes, then there are unused bytes in the end location of the IBI data buffer.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.1.45 QUEUE_THLD_CTRL

- **Name:** Queue Threshold Control Register
- **Description:** Queue Threshold Control Register is used to control thresholds that are triggering interrupts on specific thresholds of Command, Response, IBI queues. This register assumes single Command, Response and IBI queues in the Host Controller.

- **Size:** 32 bits
- **Offset:** 0x310
- **Exists:** (IC_DEVICE_ROLE<5)

IBI_STATUS_THLD	31:24
IBI_DATA_THLD	23:16
RESP_BUF_THLD	15:8
CMD_EMPTY_BUF_THLD	7:0

Table 5-52 Fields for Register: QUEUE_THLD_CTRL

Bits	Name	Memory Access	Description
31:24	IBI_STATUS_THLD	R/W	<p>IBI Status Threshold Value This field controls the generation of IBI_STATUS_THLD_STAT interrupt. If this field is programmed to 0, then the IBI_STATUS_THLD_STAT interrupt gets generated when the outstanding IBI status count in the IBI queue reaches 1 or above. If this field is programmed to N, then the IBI_STATUS_THLD_STAT interrupt gets generated when the outstanding IBI status count in the IBI queue reaches N+1 or above. Note : The valid value is only 0 if IBI with payload is selected in the configuration.</p> <p>Each IBI status entry can represent the complete (IBI payload byte size <= 4*IBI_DATA_THLD) IBI payload or a segment (IBI payload byte size > 4*IBI_DATA_THLD) of the IBI payload.</p> <p>Value After Reset: IC_DFLT_IBI_STS_THLD Exists: Always</p>
23:16	IBI_DATA_THLD	R/W	<p>IBI Data Threshold Value This field represents the IBI data segment size in Dwords (4 bytes). The minimum supported segment size is 1 (4 bytes) and the maximum supported size is IC_IBI_DATA_BUF_DEPTH-1. The IBI_DATA_THLD field enables the slicing of the incoming IBI data and generate individual status and thereby promotes the cut-through operation in reading out the IBI data.</p> <p>Value After Reset: IC_DFLT_IBI_BUF_THLD Exists: IC_HAS_IBI_DATA==1</p>
15:8	RESP_BUF_THLD	R/W	<p>Response Buffer Threshold Value. Controls the number of entries (or above) in the Response Queue that trigger the RESP_READY_STAT_INTR interrupt. The valid range is 0 to IC_RESP_BUF_DEPTH-1. The software programs only valid values. A value of 0 sets the threshold for 1 entry, and a value of N sets the threshold for N+1 entries.</p> <p>Value After Reset: IC_DFLT_RESP_BUF_THLD Exists: Always</p>

Table 5-52 Fields for Register: QUEUE_THLD_CTRL (Continued)

Bits	Name	Memory Access	Description
7:0	CMD_EMPTY_BUF_THLD	R/W	<p>Command Buffer Empty Threshold Value Controls the number of empty locations (or above) in the Command Queue that trigger CMD_QUEUE_READY_STAT interrupt. The valid range is 0 to IC_CMD_BUF_DEPTH-1. The software programs only valid values. Value of N ranging from 1 to IC_CMD_BUF_DEPTH-1 sets the threshold to N empty locations and a value of 0 sets the threshold to indicate that the queue is completely empty.</p> <p>Value After Reset: IC_DFLT_CMD_BUF_THLD Exists: Always</p>

5.1.46 DATA_BUFFER_THLD_CTRL

- **Name:** Data Buffer Threshold Control Register
- **Description:** Data Buffer Threshold Control Register used to control thresholds that are triggering interrupts on specific thresholds of Command, Response, Rx or Tx Data Buffer Queues.

- **Size:** 32 bits
- **Offset:** 0x314
- **Exists:** (IC_DEVICE_ROLE<5)

RX_START_THLD	x:24
TX_START_THLD	x:16
RX_BUF_THLD	x:8
TX_BUF_THLD	x:0

Table 5-53 Fields for Register: DATA_BUFFER_THLD_CTRL

Bits	Name	Memory Access	Description
x:24	RX_START_THLD	R/W	<p>Receive Start Threshold Value</p> <p>When the controller is set up to initiate a read transfer, it waits until the programmed number of empty locations(or more) are available in its receive buffer before it initiates the read transfer on the I3C Interface.</p> <p>The following configurable options are provided:</p> <ul style="list-style-type: none"> ■ Store and Forward Mode: If the threshold value is set to buffer size, then the controller waits for one of the following to be true to initiate the read command: <ul style="list-style-type: none"> -- Entire Receive FIFO to be empty if the data length is more than buffer size -- The data length number of locations to be empty in the Receive FIFO if data length is smaller than the buffer size. ■ Threshold Mode: In this case, if the threshold value is less than buffer size, then the controller initiates the read command as soon as the programmed locations are empty in the Receive FIFO. <p>The supported values for RX_START_THLD are:</p> <ul style="list-style-type: none"> ■ 000 - 1 ■ 001 - 4 ■ 010 - 8 ■ 011 - 16 ■ 100 - 32 ■ 101 - 64 ■ 110 - 128 ■ 111 - 256 <p>Value After Reset: IC_DFLT_RX_START_THLD Exists: Always Range Variable[x]: IC_THLD_BIT_WD + 23</p>

Table 5-53 Fields for Register: DATA_BUFFER_THLD_CTRL (Continued)

Bits	Name	Memory Access	Description
x:16	TX_START_THLD	R/W	<p>Transfer Start Threshold Value</p> <p>When the controller is set up to initiate a write transfer, it waits until the programmed number of entries (or more) are available in its transmit buffer before it initiates the write transfer on the I3C Interface.</p> <p>The following configurable options are provided:</p> <ul style="list-style-type: none"> ■ Store and Forward Mode: If the threshold value is set to buffer size, then the controller waits for one of the following to be true to initiate the write command: <ul style="list-style-type: none"> -- Entire Transmit FIFO to be full if the data length is more than buffer size -- The data length number of locations are filled in the Transmit FIFO if data length is smaller than the buffer size. ■ Threshold Mode: In this case, if the threshold value is less than buffer size, then the controller initiates the write command as soon as the programmed locations are filled in the Transmit FIFO. <p>The supported values for TX_START_THLD are:</p> <ul style="list-style-type: none"> ■ 000: 1 ■ 001: 4 ■ 010: 8 ■ 011: 16 ■ 100: 32 ■ 101: 64 ■ 110: 128 ■ 111: 256 <p>Value After Reset: IC_DFLT_TX_START_THLD Exists: Always Range Variable[x]: IC_THLD_BIT_WD + 15</p>

Table 5-53 Fields for Register: DATA_BUFFER_THLD_CTRL (Continued)

Bits	Name	Memory Access	Description
x:8	RX_BUF_THLD	R/W	<p>Receive Buffer Threshold Value.</p> <p>Controls the number of entries (or above) in the Receive FIFO that trigger the RX_THLD_STAT interrupt.</p> <p>If the programmed value is greater than the buffer depth, then threshold is set to IC_RX_BUF_DEPTH. The supported values for RX_BUF_THLD are</p> <ul style="list-style-type: none"> ■ 000: 1 ■ 001: 4 ■ 010: 8 ■ 011: 16 ■ 100: 32 ■ 101: 64 ■ 110: 128 ■ 111: 256 <p>Value After Reset: IC_DFLT_RX_BUF_THLD Exists: Always Range Variable[x]: IC_THLD_BIT_WD + 7</p>

Table 5-53 Fields for Register: DATA_BUFFER_THLD_CTRL (Continued)

Bits	Name	Memory Access	Description
x:0	TX_BUF_THLD	R/W	<p>Transmit Buffer Threshold Value.</p> <p>Controls the number of empty locations (or above) in the Transmit FIFO that trigger the TX_THLD_STAT interrupt. If the programmed value is greater than the buffer depth, then threshold is set to IC_TX_BUF_DEPTH. The supported values for TX_BUF_THLD are</p> <ul style="list-style-type: none"> ■ 000: 1 ■ 001: 4 ■ 010: 8 ■ 011: 16 ■ 100: 32 ■ 101: 64 ■ 110: 128 ■ 111: 256 <p>Value After Reset: IC_DFLT_TX_BUF_THLD Exists: Always Range Variable[x]: IC_THLD_BIT_WD - 1</p>

5.1.47 QUEUE_SIZE_CTRL

- **Name:** Queue Size Register
- **Description:** Command Queue, Response Queue, Data buffer, IBI status queue sizes.
- **Size:** 32 bits
- **Offset:** 0x318
- **Exists:** (IC_DEVICE_ROLE<5)

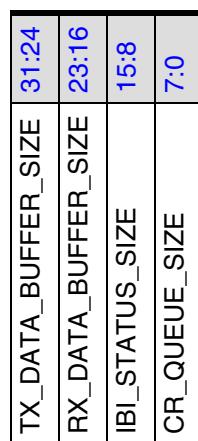


Table 5-54 Fields for Register: QUEUE_SIZE_CTRL

Bits	Name	Memory Access	Description
31:24	TX_DATA_BUFFER_SIZE	R	Transmit Data sing size Value After Reset: IC_TX_DATA_BUF_SIZE Exists: Always
23:16	RX_DATA_BUFFER_SIZE	R	Receive Data ring size Value After Reset: IC_RX_DATA_BUF_SIZE Exists: Always
15:8	IBI_STATUS_SIZE	R	IBI ring size Value After Reset: IC_IBI_BUF_DEPTH Exists: Always

Table 5-54 Fields for Register: QUEUE_SIZE_CTRL (Continued)

Bits	Name	Memory Access	Description
7:0	CR_QUEUE_SIZE	R	<p>Command/Response ring size.</p> <p>Value After Reset: IC_CMD_BUF_DEPTH</p> <p>Exists: Always</p>

5.1.48 PIO_INTR_STATUS

- **Name:** PIO Interrupt Status Register
- **Description:** The PIO Interrupt Status register indicates the status of outstanding interrupts.

- **Size:** 32 bits
- **Offset:** 0x320
- **Exists:** (IC_DEVICE_ROLE<5)

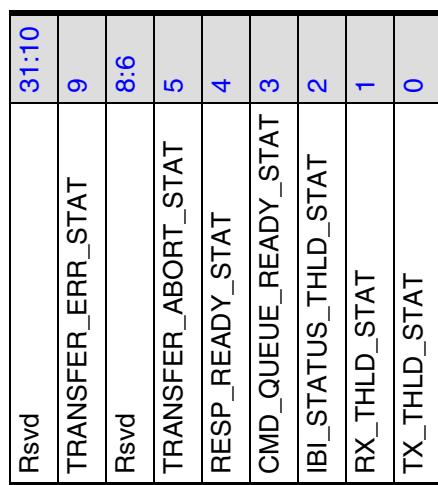


Table 5-55 Fields for Register: PIO_INTR_STATUS

Bits	Name	Memory Access	Description
31:10			Reserved Field: Yes
9	TRANSFER_ERR_STAT	R/W1C	Transfer Error Status The Host Controller sets this bit to 1'b1 when any transfer error occurs on the I3C Bus. The Error Type for this error is available in the Response structure corresponding to this transfer/command. To clear, write 1'b0 to this bit. Value After Reset: 0x0 Exists: Always
8:6			Reserved Field: Yes
5	TRANSFER_ABORT_STAT	R/W1C	Transfer Abort Status The Host Controller sets this bit to 1'b1 when any transfer is aborted. To clear, write 1'b0 to this bit. Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4

Table 5-55 Fields for Register: PIO_INTR_STATUS (Continued)

Bits	Name	Memory Access	Description
4	RESP_READY_STAT	R	<p>Response Ready Status The Host Controller sets this bit to 1'b1 when the number of Response Queue entries is \geq the RESP_BUF_THLD threshold (See Register QUEUE_THLD_CTRL). The Host Controller automatically clears this field to 1'b0 when the number of Response Queue entries falls below the RESP_BUF_THLD threshold.</p> <p>Value After Reset: 0x0 Exists: Always</p>
3	CMD_QUEUE_READY_STAT	R	<p>Command Queue Ready Status The Host Controller sets this bit to 1'b1 when the number of Command Queue entries is \leq the CMD_EMPTY_BUF_THLD threshold (See Register QUEUE_THLD_CTRL). The Host Controller automatically clears this field to 1'b0 when the number of Command Queue entries exceeds the RESP_BUF_THLD threshold.</p> <p>Value After Reset: 0x0 Exists: Always</p>
2	IBI_STATUS_THLD_STAT	R	<p>IBI Status Threshold Status The Host Controller sets this bit to 1'b1 when the number of IBI Status Entries in the IBI Queue reaches the IBI_STATUS_THLD threshold (See Register QUEUE_THLD_CTRL). The Host Controller automatically clears this field to 1'b0 when the number of IBI Status entries in the IBI Queue falls below the IBI_STATUS_THLD threshold as a result of application reads.</p> <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4</p>
1	RX_THLD_STAT	R	<p>Rx Data Buffer Threshold Status The Host Controller sets this bit to 1'b1 when the number of entries in the Rx Data Queue is \geq the RX_BUF_THLD threshold (See Register DATA_BUFFER_THLD_CTRL). The Host Controller automatically clears this field to 1'b0 when the number of Rx Data Queue entries falls below the RX_BUF_THLD threshold.</p> <p>Value After Reset: 0x0 Exists: Always</p>
0	TX_THLD_STAT	R	<p>Tx Data Buffer Threshold Status The Host Controller sets this bit to 1'b1 when the number of entries in the Tx Data Queue is \leq the TX_BUF_THLD threshold (see register DATA_BUFFER_THLD_CTRL). The Host Controller automatically clears this field to 1'b0 when the number of Tx Data Queue entries exceeds the TX_BUF_THLD threshold.</p> <p>Value After Reset: 0x0 Exists: Always</p>

5.1.49 PIO_INTR_STATUS_ENABLE

- **Name:** PIO Interrupt Status Enable Register
- **Description:** The PIO Interrupt Status Enable register enables reporting of outstanding interrupts.
- **Size:** 32 bits
- **Offset:** 0x324
- **Exists:** (IC_DEVICE_ROLE<5)

31:10	Rsvd	TRANSFER_ERR_STAT_EN	9	Rsvd	TRANSFER_ABORT_STAT_EN	5	RESP_READY_STAT_INTR_EN	4	CMD_QUEUE_READY_STAT_EN	3	IBI_THLD_STAT_EN	2	RX_THLD_STAT_EN	1	TX_THLD_STAT_EN	0
-------	------	----------------------	---	------	------------------------	---	-------------------------	---	-------------------------	---	------------------	---	-----------------	---	-----------------	---

Table 5-56 Fields for Register: PIO_INTR_STATUS_ENABLE

Bits	Name	Memory Access	Description
31:10			Reserved Field: Yes
9	TRANSFER_ERR_STAT_EN	R/W	When set to 1'b1, enables interrupt status logging for TRANSFER_ERR_STAT. Value After Reset: 0x0 Exists: Always
8:6			Reserved Field: Yes
5	TRANSFER_ABORT_STAT_EN	R/W	When set to 1'b1, enables interrupt status logging for TRANSFER_ABORT_STAT. Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4

Table 5-56 Fields for Register: PIO_INTR_STATUS_ENABLE (Continued)

Bits	Name	Memory Access	Description
4	RESP_READY_STAT_INTR_EN	R/W	<p>When set to 1'b1, enables interrupt status logging for RESP_READY_STAT.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
3	CMD_QUEUE_READY_STAT_EN	R/W	<p>When set to 1'b1, enables interrupt status logging for CMD_QUEUE_READY_STAT.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
2	IBI_THLD_STAT_EN	R/W	<p>When set to 1'b1, enables interrupt status logging for IBI_STATUS_THLD_STAT.</p> <p>Value After Reset: 0x0</p> <p>Exists: IC_DEVICE_ROLE<4</p>
1	RX_THLD_STAT_EN	R/W	<p>When set to 1'b1, enables interrupt status logging for RX_THLD_STAT.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
0	TX_THLD_STAT_EN	R/W	<p>When set to 1'b1, enables interrupt status logging for TX_THLD_STAT</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.1.50 PIO_INTR_SIGNAL_ENABLE

- **Name:** PIO Interrupt Signal Enable Register
- **Description:** The PIO Interrupt Signal Enable register enables signaling of outstanding interrupts received by the Host Controller.
- **Size:** 32 bits
- **Offset:** 0x328
- **Exists:** (IC_DEVICE_ROLE<5)

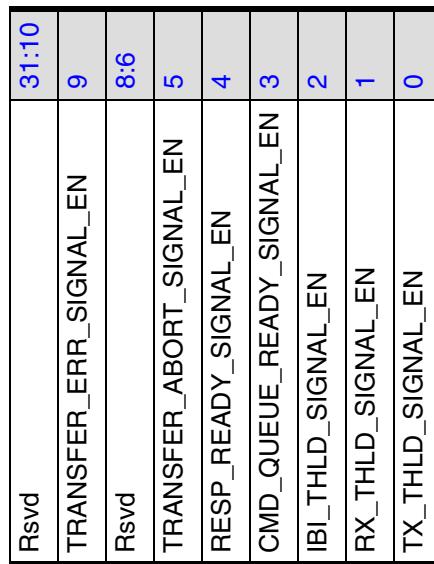


Table 5-57 Fields for Register: PIO_INTR_SIGNAL_ENABLE

Bits	Name	Memory Access	Description
31:10			Reserved Field: Yes
9	TRANSFER_ERR_SIGNAL_EN	R/W	When set to 1'b1 & field TRANSFER_ERR_STAT is set, asserts interrupt to Host. Value After Reset: 0x0 Exists: Always
8:6			Reserved Field: Yes
5	TRANSFER_ABORT_SIGNAL_EN	R/W	When set to 1'b1 & field TRANSFER_ABORT_STAT is set, asserts interrupt to Host. Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4

Table 5-57 Fields for Register: PIO_INTR_SIGNAL_ENABLE (Continued)

Bits	Name	Memory Access	Description
4	RESP_READY_SIGNAL_EN	R/W	<p>When set to 1'b1 & field RESP_READY_STAT is set, asserts interrupt to Host.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
3	CMD_QUEUE_READY_SIGNAL_EN	R/W	<p>When set to 1'b1 & field CMD_QUEUE_READY_STAT is set, asserts interrupt to Host.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
2	IBI_THLD_SIGNAL_EN	R/W	<p>When set to 1'b1 & field IBI_STATUS_THLD_STAT is set, asserts interrupt to Host.</p> <p>Value After Reset: 0x0</p> <p>Exists: IC_DEVICE_ROLE<4</p>
1	RX_THLD_SIGNAL_EN	R/W	<p>When set to 1'b1 & field RX_THLD_STAT is set, asserts interrupt to Host.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
0	TX_THLD_SIGNAL_EN	R/W	<p>When set to 1'b1 & field TX_THLD_STAT is set, asserts interrupt to Host.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.1.51 PIO_INTR_FORCE

- **Name:** PIO Interrupt Force Register
- **Description:** The PIO Interrupt Force register is used to force specific interrupt. It can be used for debug purposes.
- **Size:** 32 bits
- **Offset:** 0x32c
- **Exists:** (IC_DEVICE_ROLE<5)

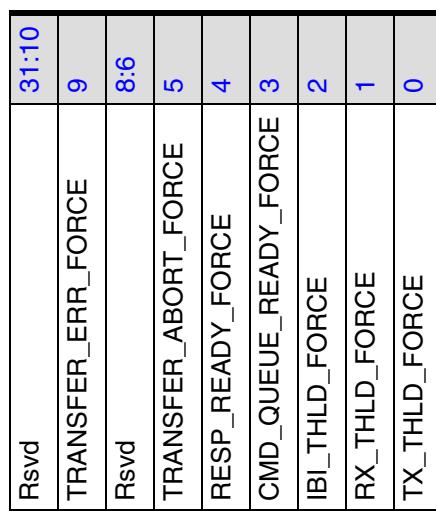


Table 5-58 Fields for Register: PIO_INTR_FORCE

Bits	Name	Memory Access	Description
31:10			Reserved Field: Yes
9	TRANSFER_ERR_FORCE	W	For software testing, when TRANSFER_ERR_FORCE set to 1'b1, forces the corresponding interrupt, subject to INTR_STAT_EN and INTR_SIGNAL_EN configuration. Value After Reset: 0x0 Exists: Always
8:6			Reserved Field: Yes
5	TRANSFER_ABORT_FORCE	W	For software testing, when TRANSFER_ABORT_FORCE set to 1'b1, forces the corresponding interrupt, subject to INTR_STAT_EN and INTR_SIGNAL_EN configuration. Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4

Table 5-58 Fields for Register: PIO_INTR_FORCE (Continued)

Bits	Name	Memory Access	Description
4	RESP_READY_FORCE	W	<p>For software testing, when RESP_READY_FORCE set to 1'b1, forces the corresponding interrupt, subject to INTR_STAT_EN and INTR_SIGNAL_EN configuration.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
3	CMD_QUEUE_READY_FORCE	W	<p>For software testing, when CMD_QUEUE_READY_FORCE set to 1'b1, forces the corresponding interrupt, subject to INTR_STAT_EN and INTR_SIGNAL_EN configuration.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
2	IBI_THLD_FORCE	W	<p>For software testing, when IBI_THLD_FORCE set to 1'b1, forces the corresponding interrupt, subject to INTR_STAT_EN and INTR_SIGNAL_EN configuration.</p> <p>Value After Reset: 0x0</p> <p>Exists: IC_DEVICE_ROLE<4</p>
1	RX_THLD_FORCE	W	<p>For software testing, when RX_THLD_FORCE set to 1'b1, forces the corresponding interrupt, subject to INTR_STAT_EN and INTR_SIGNAL_EN configuration.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
0	TX_THLD_FORCE	W	<p>For software testing, when TX_THLD_FORCE set to 1'b1, forces the corresponding interrupt, subject to INTR_STAT_EN and INTR_SIGNAL_EN configuration.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.1.52 DEV_ADDR_TABLE1_LOC1

- **Name:** Device Address Table Location 1 of Device1
- **Description:** Device Address Table Location 1 of Device1 1
- **Size:** 32 bits
- **Offset:** 0x400
- **Exists:** 2 <= IC_DEV_ADDR_TABLE_BUF_DEPTH && IC_DEVICE_ROLE<4 && IC_HAS_HCI==1 && IC_HAS_DAT==1

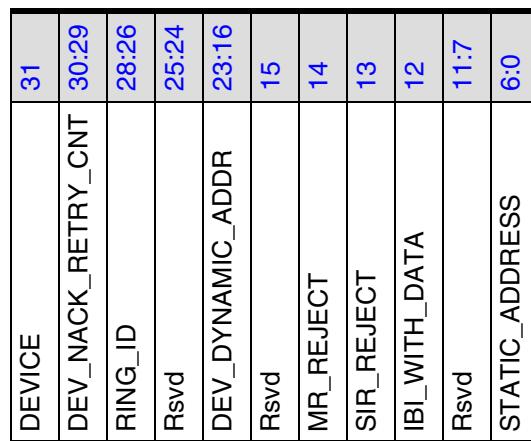


Table 5-59 Fields for Register: DEV_ADDR_TABLE1_LOC1

Bits	Name	Memory Access	Description
31	DEVICE	R/W	<p>Type of device</p> <ul style="list-style-type: none"> ■ 0: I3C ■ 1: I2C <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>

Table 5-59 Fields for Register: DEV_ADDR_TABLE1_LOC1 (Continued)

Bits	Name	Memory Access	Description
30:29	DEV_NACK_RETRY_CNT	R/W	<p>This field is used to set the Device NACK Retry count for the particular device.</p> <p>If the Device NACK's for the device address, the controller automatically retries the same device until this count expires. If the Slave does not ACK for the mentioned number of retries, then Controller generates an error response and move to the Halt state.</p> <p>This feature is used for Retry Model for the following features mentioned in the I3C Specification:</p> <ul style="list-style-type: none"> ■ Retry Model for Direct GET CCC Commands. ■ The incoming SIR-IBI matches with the slave address initiated by the Master. <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
28:26	RING_ID	R/W	<p>Ring group identification. This field is used to put IBI from specific device to appropriate ring bundle.</p> <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
25:24			Reserved Field: Yes
23:16	DEV_DYNAMIC_ADDR	R/W	<p>Device Dynamic Address with parity. The MSB, bit[23], should be programmed with parity of dynamic address.</p> <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
15			Reserved Field: Yes

Table 5-59 Fields for Register: DEV_ADDR_TABLE1_LOC1 (Continued)

Bits	Name	Memory Access	Description
14	MR_REJECT	R/W	<p>In-Band Master Request Reject field is used to control, per device, whether to accept Master request from Devices.</p> <ul style="list-style-type: none"> ■ 0x0 - Accept: ACK the Master Request ■ 0x1 - Reject: NACK the Master Request and send auto disable CCC. <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
13	SIR_REJECT	R/W	<p>In-Band Slave Interrupt Request Reject field is used to control, per device, whether to accept Slave Interrupt request from Devices.</p> <ul style="list-style-type: none"> ■ 0x0 - Accept: ACK the SIR ■ 0x1 - Reject: NACK the SIR and send auto disable CCC. <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
12	IBI_WITH_DATA	R/W	<p>Mandatory one or more data bytes follow the accepted IBI from the device. Data byte continuation is indicated by T-Bit.</p> <ul style="list-style-type: none"> ■ 0x0 - IBI Without Mandatory Byte ■ 0x1 - IBI with one or more Mandatory Bytes. <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
11:7			Reserved Field: Yes
6:0	STATIC_ADDRESS	R/W	<p>Device Static Address.</p> <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>

5.1.53 DEV_ADDR_TABLE1_LOC2

- **Name:** Device Address Table Location 2 of Device1
- **Description:** Device Address Table Location 2 of Device1 1
- **Size:** 32 bits
- **Offset:** 0x404
- **Exists:** 2 <= IC_DEV_ADDR_TABLE_BUF_DEPTH && IC_DEVICE_ROLE<4 && IC_HAS_HCI==1 && IC_HAS_DAT==1

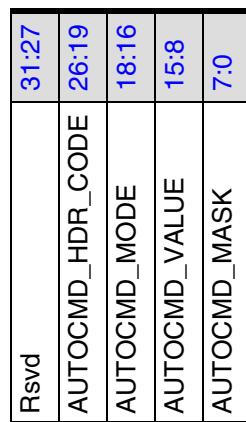


Table 5-60 Fields for Register: DEV_ADDR_TABLE1_LOC2

Bits	Name	Memory Access	Description
31:27			Reserved Field: Yes
26:19	AUTOCMD_HDR_CODE	R/W	Auto command HDR code Value After Reset: 0x0 Exists: Always Testable: untestable
18:16	AUTOCMD_MODE	R/W	Mode of automatic Read transaction on the Bus (Auto Command feature). Value After Reset: 0x0 Exists: Always Testable: untestable

Table 5-60 Fields for Register: DEV_ADDR_TABLE1_LOC2 (Continued)

Bits	Name	Memory Access	Description
15:8	AUTOCMD_VALUE	R/W	<p>Value of IBI mandatory byte that triggers automatic Read transaction on the Bus(Auto Command feature).</p> <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
7:0	AUTOCMD_MASK	R/W	<p>Mask of IBI mandatory byte that triggers automatic Read transaction on the Bus(Auto Command feature).</p> <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>

5.1.54 DEV_CHAR_TABLE1_LOC1

- **Description:** Device Characteristic Table Location-1 of Device1 1
- **Size:** 32 bits
- **Offset:** 0x600
- **Exists:** $4 \leq \text{IC_DEV_CHAR_TABLE_BUF_DEPTH} \&\& \text{IC_DEVICE_ROLE} < 4 \&\& \text{IC_HAS_HCI} == 1 \&\& \text{IC_HAS_DAT} == 1$



Table 5-61 Fields for Register: DEV_CHAR_TABLE1_LOC1

Bits	Name	Memory Access	Description
31:0	MSB_PROVISIONAL_ID	R	<p>The LSB 32-bit value of Provisional-ID Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true</p>

5.1.55 DEV_CHAR_TABLE1_LOC2

- **Description:** Device Characteristic Table Location-2 of Device1 1
- **Size:** 32 bits
- **Offset:** 0x604
- **Exists:** $4 \leq \text{IC_DEV_CHAR_TABLE_BUF_DEPTH} \&\& \text{IC_DEVICE_ROLE} < 4 \&\& \text{IC_HAS_HCI} == 1 \&\& \text{IC_HAS_DAT} == 1$



Table 5-62 Fields for Register: DEV_CHAR_TABLE1_LOC2

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15:0	LSB_PROVISIONAL_ID	R	The MSB 16-bit value of Provisional-ID Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true

5.1.56 DEV_CHAR_TABLE1_LOC3

- **Description:** Device Characteristic Table Location-3 of Device1 1
- **Size:** 32 bits
- **Offset:** 0x608
- **Exists:** 4 <= IC_DEV_CHAR_TABLE_BUF_DEPTH && IC_DEVICE_ROLE<4 && IC_HAS_HCI==1 && IC_HAS_DAT==1

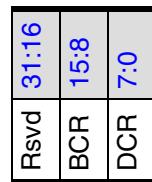


Table 5-63 Fields for Register: DEV_CHAR_TABLE1_LOC3

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15:8	BCR	R	Bus Characteristic Value Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true
7:0	DCR	R	Device Characteristic Value Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true

5.1.57 DEV_CHAR_TABLE1_LOC4

- **Description:** Device Characteristic Table Location-4 of Device1 1
- **Size:** 32 bits
- **Offset:** 0x60c
- **Exists:** $4 \leq IC_DEV_CHAR_TABLE_BUF_DEPTH \&\& IC_DEVICE_ROLE < 4 \&\& IC_HAS_HCI == 1 \&\& IC_HAS_DAT == 1$



Table 5-64 Fields for Register: DEV_CHAR_TABLE1_LOC4

Bits	Name	Memory Access	Description
31:8			Reserved Field: Yes
7:0	DEV_DYNAMIC_ADDR	R	Device Dynamic Address assigned. Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true

5.2 DWC_mipi_i3c_map/DWC_mipi_i3c_block Registers

DWC_mipi_i3c . Follow the link for the register to see a detailed description of the register.

Table 5-65 Registers for Address Block: DWC_mipi_i3c_map/DWC_mipi_i3c_block

Register	Offset	Description
DEVICE_CTRL on page 313	0x0	Device Control Register
DEVICE_ADDR on page 319	0x4	Device Address Register
HW_CAPABILITY on page 323	0x8	Hardware Capability register
COMMAND_QUEUE_PORT on page 326	0xc	COMMAND_QUEUE_PORT
RESPONSE_QUEUE_PORT on page 327	0x10	RESPONSE_QUEUE_PORT
RX_DATA_PORT on page 328	0x14	Receive Data Port Register
TX_DATA_PORT on page 329	0x14	Transmit Data Port Register
IBI_QUEUE_DATA on page 330	0x18	In-Band Interrupt Queue Data Register
IBI_QUEUE_STATUS on page 331	0x18	In-Band Interrupt Queue Status Register
QUEUE_THLD_CTRL on page 333	0x1c	Queue Threshold Control Register
DATA_BUFFER_THLD_CTRL on page 336	0x20	Data Buffer Threshold Control Register
IBI_QUEUE_CTRL on page 340	0x24	IBI Queue Control Register
IBI_MR_REQ_REJECT on page 343	0x2c	IBI MR Request Rejection Control Register
IBI_SIR_REQ_REJECT on page 344	0x30	IBI SIR Request Rejection Control Register
RESET_CTRL on page 345	0x34	Reset Control Register
SLV_EVENT_STATUS on page 348	0x38	Slave Event Status Register
INTR_STATUS on page 351	0x3c	Interrupt Status Register
INTR_STATUS_EN on page 356	0x40	Interrupt Status Enable Register
INTR_SIGNAL_EN on page 359	0x44	Interrupt Signal Enable Register
INTR_FORCE on page 362	0x48	Interrupt Force Enable Register
QUEUE_STATUS_LEVEL on page 365	0x4c	Queue Status Level Register
DATA_BUFFER_STATUS_LEVEL on page 367	0x50	Data Buffer Status Level Register
PRESENT_STATE on page 368	0x54	Present State Register
CCC_DEVICE_STATUS on page 372	0x58	Device Operating Status Register

Table 5-65 Registers for Address Block: DWC_mipi_i3c_map/DWC_mipi_i3c_block (Continued)

Register	Offset	Description
DEVICE_ADDR_TABLE_POINTER on page 375	0x5c	Pointer for Device Address Table Registers
DEV_CHAR_TABLE_POINTER on page 377	0x60	Pointer for Device Characteristics Table
VENDOR_SPECIFIC_REG_POINTER on page 379	0x6c	Pointer for Vendor specific Registers
SLV_MIPI_ID_VALUE on page 380	0x70	Provisional ID Register
SLV_PID_VALUE on page 381	0x74	Provisional ID Register
SLV_CHAR_CTRL on page 383	0x78	I3C Slave Characteristic Register
SLV_MAX_LEN on page 386	0x7c	I3C Max Write/Read Length Register
MAX_READ_TURNAROUND on page 387	0x80	MXDS Maximum Read Turnaround Time Register
MAX_DATA_SPEED on page 388	0x84	MXDS Maximum Data Speed Register
SLV_INTR_REQ on page 391	0x8c	Slave Interrupt Request Register
SLV_TSX_SYMBL_TIMING on page 394	0x90	TSP/TSL Symbol Timing Register
DEVICE_CTRL_EXTENDED on page 395	0xb0	Device Control Extended Register
SCL_I3C_OD_TIMING on page 397	0xb4	SCL I3C Open Drain Timing Register
SCL_I3C_PP_TIMING on page 398	0xb8	SCL I3C Push Pull Timing Register
SCL_I2C_FM_TIMING on page 399	0xbc	SCL I2C Fast Mode Timing Register
SCL_I2C_FMP_TIMING on page 400	0xc0	SCL I2C Fast Mode Plus Timing Register
SCL_EXT_LCNT_TIMING on page 401	* Varies	SCL Extended Low Count Timing Register
SCL_EXT_TERMN_LCNT_TIMING on page 403	* Varies	SCL Termination Bit Low count Timing Register
SDA_HOLD_SWITCH_DLY_TIMING on page 405	* Varies	SDA Hold and Mode Switch Delay Timing Register
BUS_FREE_AVAIL_TIMING on page 407	* Varies	Bus Free Timing Register
BUS_IDLE_TIMING on page 409	* Varies	Bus Idle Timing Register
SCL_LOW_MST_EXT_TIMEOUT on page 410	* Varies	SCL_LOW_MST_TIMEOUT
I3C_VER_ID on page 411	0xe0	DWC_mipi_i3c Version ID Register
I3C_VER_TYPE on page 412	0xe4	DWC_mipi_i3c Version Type Register

Table 5-65 Registers for Address Block: DWC_mipi_i3c_map/DWC_mipi_i3c_block (Continued)

Register	Offset	Description
QUEUE_SIZE_CAPABILITY on page 413	0xe8	DWC_mipi_i3c Queue Size Capability Register
DEV_CHAR_TABLE1_LOC1 on page 416	* Varies	Device Characteristic Table Location-1 of Device1 This register is used in master mode of...
SEC_DEV_CHAR_TABLE1 on page 417	* Varies	Secondary Master Device Characteristic Table Location of Device1 1
DEV_CHAR_TABLE1_LOC2 on page 418	* Varies	Device Characteristic Table Location-2 of Device1 This register is used in master mode of...
DEV_CHAR_TABLE1_LOC3 on page 419	* Varies	Device Characteristic Table Location-3 of Device1 This register is used in master mode of...
DEV_CHAR_TABLE1_LOC4 on page 420	* Varies	Device Characteristic Table Location-4 of Device1 This register is used in master mode of...
DEV_ADDR_TABLE1_LOC1 on page 421	* Varies	Device Address Table Location of Device1
DEV_ADDR_TABLE_LOC1 on page 424	* Varies	Device Address Table of Device1

5.2.1 DEVICE_CTRL

- **Name:** Device Control Register
- **Description:** DWC_mipi_i3c control Register

This Register controls the transfer properties and disposition of controller's capabilities.

- **Size:** 32 bits
- **Offset:** 0x0
- **Exists:** (IC_DEVICE_ROLE<5)

ENABLE	31
RESUME	30
ABORT	29
DMA_ENABLE	28
ADAPTIVE_I2C_I3C	27
Rsvd	26
IDLE_CNT_MULTPLIER	25:24
Rsvd	23:9
HOT_JOIN_CTRL	8
I2C_SLAVE_PRESENT	7
Rsvd	6:1
IBA_INCLUDE	0

Table 5-66 Fields for Register: DEVICE_CTRL

Bits	Name	Memory Access	Description
31	ENABLE	R/W	<p>Controls whether or not DWC_mipi_i3c is enabled.</p> <ul style="list-style-type: none"> ■ 1: Enables the DWC_mipi_i3c controller. ■ 0: Disables the DWC_mipi_i3c controller. <p>In Master mode of operation, software can disable DWC_mipi_i3c while it is active. However, the controller may not get disabled immediately and is 'Disabled' after commands in the Command queue (if any) are executed leading to a STOP condition on the bus and Master FSM is in IDLE state (as indicated by PRESENT_STATE Register).</p> <p>In Slave mode of operation, software can disable DWC_mipi_i3c while it is active. However, the disable happens after the ongoing transfer is completed on the I3C bus. Software can read back 1'b0 from this field once disabling of DWC_mipi_i3c is completed. After power on reset, the software can enable I3C slave controller by programming this bit to 1'b1. However, the I3C bus interface of the controller, responds to transfer on the bus only after it observes Bus Available condition for $BUS_AVAILABLE_TIME * IDLE_CNT_MULTPLIER$ counts of pclk period. The successful completion of Enable/Disable of the controller depends on availability of SCL to the controller at the time of performing this operation, and hence may not happen instantly.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>

Table 5-66 Fields for Register: DEVICE_CTRL (Continued)

Bits	Name	Memory Access	Description
30	RESUME	R/W	<p>DWC_mipi_i3c Resume This bit is used to resume the controller after it goes to the halt state.</p> <p>In the Master mode of operation, the controller goes to the halt state (as indicated in PRESENT_STATE Register) due to any type of error in the transfer (the type of error is indicated by ERR_STATUS field in the RESPONSE_QUEUE_PORT register). After the controller goes to the halt state, the application has to write 1'b1 to this bit to resume the controller. This bit is auto-cleared once the controller resumes the transfers by initiating the next command.</p> <p>In the Slave mode of operation, the controller goes to the halt state due to following conditions:</p> <ul style="list-style-type: none"> ■ Any type of error in the transfer (the type of error is indicated by ERR_STATUS field in the RESPONSE_QUEUE_PORT register) ■ MRL Register updated by the master through SETMRL CCC. <p>After the controller goes to the halt state, the application has to take necessary action to handle the error condition and then write 1'b1 to this bit to resume the controller. This bit is auto-cleared once the controller is ready to accept new transfers.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>
29	ABORT	R/W	<p>DWC_mipi_i3c Abort This bit is used in Master mode of operation. This bit allows the controller to relinquish the DWC_mipi_i3c bus before completing the issued transfer. In response to an ABORT request, the controller issues the STOP condition after the complete data byte is transferred or received. This bit is auto-cleared once the transfer is aborted and the controller issues a 'Transfer Abort' interrupt.</p> <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4 Volatile: true</p>

Table 5-66 Fields for Register: DEVICE_CTRL (Continued)

Bits	Name	Memory Access	Description
28	DMA_ENABLE	R/W	<p>DMA Handshake Interface Enable This bit is used to enable or disable the DMA Handshaking interface, and is applicable to both Master and Slave modes of operation.</p> <ul style="list-style-type: none"> ■ 1: Enables the DMA handshake control to interact with external DMA. ■ 0: The DMA handshake control has no significance. <p>Value After Reset: 0x0 Exists: IC_HAS_DMA==1</p>
27	ADAPTIVE_I2C_I3C	R/W	<p>This field is used in Slave mode of operation. Note that when mode_i2c strap is driven to '0', the Slave controller operates in Adaptive Mode. Setting of this bit is NOT required to put the controller in Adaptive Mode. It is only used to enable some features of the Slave controller to adapt to "Adaptive I2C/I3C mode" of operation. This bit is cleared automatically if the controller determines the mode as I3C. Effect on Hot-Join: If this bit is programmed to 1'b1, the controller initiates a Hot-Join request only after it has switched to I3C mode of operation. If this bit is not set, the controller initiates a Hot-Join without determining the bus mode assuming itself to be on DWC_mipi_i3c bus. This bit should be set only if the Slave application does not know to which bus the device is connected to.</p> <p>Value After Reset: 0x0 Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) Volatile: true</p>
26			Reserved Field: Yes

Table 5-66 Fields for Register: DEVICE_CTRL (Continued)

Bits	Name	Memory Access	Description
25:24	IDLE_CNT_MULTIPLIER	R/W	<p>Idle Count Multiplier This bit is used in Slave mode of operation. After power-on reset, the Slave controller is enabled only after it sees both SDA and SCL lines idle for a specified time. This idle time is calculated by multiplying IDLE_CNT_MULTIPLIER with BUS_AVAILABLE_TIME field in the BUS_FREE_AVAIL_TIMING register. - 00 - BUS_AVAILABLE_TIME * 1 - 01 - BUS_AVAILABLE_TIME * 2 - 10 - BUS_AVAILABLE_TIME * 4 - 11 - BUS_AVAILABLE_TIME * 8</p> <p>Value After Reset: 0x0 Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)</p>
23:9			Reserved Field: Yes
8	HOT_JOIN_CTRL	R/W	<p>Hot-Join ACK/NACK Control This bit is used in master mode of operation. This bit acts as a global control to ACK/NACK the Hot-Join request from the devices. The DWC_mipi_i3c Master ACK/NACKs the Hot-Join request from other devices connected on the DWC_mipi_i3c bus, based on programming of this bit.</p> <ul style="list-style-type: none"> ■ 0: ACK the Hot-join request. ■ 1: NACK and send broadcast CCC to disable Hot-Join. <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): Ack Hot-Join requests ■ 0x1 (ENABLED): Nack and auto-disable Hot-Join request <p>Value After Reset: IC_DFLT_HJ_CTRL Exists: IC_DEVICE_ROLE<4</p>
7	I2C_SLAVE_PRESENT	R/W	<p>I2C Slave Present This bit is used in master mode of operation. This bit indicates whether any Legacy I2C devices are present in the system. In HDR mode, this field is used to select TSL over TSP in a mixed bus configuration.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): I2C Slave not present ■ 0x1 (ENABLED): I2C Slave present <p>Value After Reset: IC_DFLT_I2C_SLAVE_PRESENT Exists: IC_DEVICE_ROLE<4</p>

Table 5-66 Fields for Register: DEVICE_CTRL (Continued)

Bits	Name	Memory Access	Description
6:1			Reserved Field: Yes
0	IBA_INCLUDE	R/W	<p>I3C Broadcast Address include This bit is used in Master mode of operation. This bit is used to include DWC_mipi_i3c broadcast address (0x7E) for private transfer.</p> <p>Note: If DWC_mipi_i3c broadcast address is not included for the private transfers, In-band Interrupts (IBI) driven from Slaves might not win address arbitration. Hence, the IBIs get delayed.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (NOT_INCLUDED): I3C Broadcast Address is not included for Private Transfers ■ 0x1 (INCLUDED): I3C Broadcast Address is included for Private Transfers <p>Value After Reset: IC_DFLT_IBA_INC Exists: IC_DEVICE_ROLE<4</p>

5.2.2 DEVICE_ADDR

- **Name:** Device Address Register
- **Description:**

In the master mode of operation this Register is used to program the Device Dynamic Addresses and its respective valid bit.

In the slave mode of operation this Register reflects the Static and Dynamic Addresses and their respective valid bits of the slave controller.

- **Size:** 32 bits
- **Offset:** 0x4
- **Exists:** (IC_DEVICE_ROLE<5)

DYNAMIC_ADDR_VALID	31	Rsvd	30:23	DYNAMIC_ADDR	22:16	STATIC_ADDR_VALID	15	Rsvd	14:7	STATIC_ADDR	6:0
--------------------	----	------	-------	--------------	-------	-------------------	----	------	------	-------------	-----

Table 5-67 Fields for Register: DEVICE_ADDR

Bits	Name	Memory Access	Description
31	DYNAMIC_ADDR_VALID	* Varies	<p>Dynamic Address Valid This bit is used to control whether the DYNAMIC_ADDR is valid or not.</p> <ul style="list-style-type: none"> ■ In I3C Main Master mode, the user sets this bit to 1 as it self-assigns its dynamic address. ■ In all other operation modes, the Controller sets this bit to 1 when Main Master assigns the Dynamic address during ENTDAA or SETDASA mechanism. ■ In I3C Slave Mode the Controller sets this bit to 1 when Main Master assigns the Dynamic address during ENTDAA or SETDASA mechanism. <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (INVALID): Dynamic Address is invalid ■ 0x1 (VALID): Dynamic Address is valid <p>Value After Reset: IC_DFLT_DYNAMIC_ADDR_VALID</p> <p>Exists: Always</p> <p>Volatile: true</p> <p>Memory Access: "(IC_DEVICE_ROLE<4) ? \"read-write\" : \"read-only\""</p>
30:23			Reserved Field: Yes

Table 5-67 Fields for Register: DEVICE_ADDR (Continued)

Bits	Name	Memory Access	Description
22:16	DYNAMIC_ADDR	* Varies	<p>Device Dynamic Address. This field is used to program the Device Dynamic Address. The Controller uses this address for I3C transfers.</p> <ul style="list-style-type: none"> ■ In Main Master mode, the user/application has to program the Dynamic Address through the Slave interface as it self-assigns its Dynamic Address. ■ In all other modes, the Main Master assigns this address during ENTDA or SETDASA mechanism. ■ The Main Master assigns this address during ENTDA or SETDASA mechanism. <p>Value After Reset: IC_DFLT_DYNAMIC_ADDR Exists: Always Volatile: true Memory Access: "(IC_DEVICE_ROLE<4) ? \"read-write\" : \"read-only\""</p>
15	STATIC_ADDR_VALID	R/W	<p>Static Address Valid. In slave mode of operation this bit reflects the value of static_addr_en input port. The input port static_addr_en is expected to be driven to 1 only if the device supports I2C or I3C Static Address.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (INVALID): Static Address is invalid ■ 0x1 (VALID): Static Address is valid <p>Value After Reset: 0x0 Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) Volatile: true</p>
14:7			Reserved Field: Yes

Table 5-67 Fields for Register: DEVICE_ADDR (Continued)

Bits	Name	Memory Access	Description
6:0	STATIC_ADDR	R/W	<p>Device Static Address. In slave mode of operation this field reflects the value of static_addr input port. The controller uses this address to respond to SETDASA CCC Command to get the Dynamic Address if static address is valid (static_addr_en port is set to 1).</p> <p>Value After Reset: 0x0 Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) Volatile: true</p>

5.2.3 HW_CAPABILITY

- **Name:** Hardware Capability register
- **Description:** Hardware Capability register

This register reflects the configured capabilities of DWC_mipi_i3c.

- **Size:** 32 bits
- **Offset:** 0x8
- **Exists:** Always

Rsvd	31:20	SLV_IBI_CAP	19	SLV_HJ_CAP	18	DMA_EN	17	HDR_TX_CLOCK_PERIOD	16:11	CLOCK_PERIOD	10:5	HDR_TS_EN	4	HDR_DDR_EN	3	DEVICE_ROLE_CONFIG	2:0
------	-------	-------------	----	------------	----	--------	----	---------------------	-------	--------------	------	-----------	---	------------	---	--------------------	-----

Table 5-68 Fields for Register: HW_CAPABILITY

Bits	Name	Memory Access	Description
31:20			Reserved Field: Yes
19	SLV_IBI_CAP	R	Reflects the IC_SLV_IBI Configurable Parameter. Specifies slave's capability to initiate slave interrupt requests. Value After Reset: IC_SLV_IBI Exists: Always
18	SLV_HJ_CAP	R	Reflects the IC_SLV_HJ Configurable Parameter. Specifies slave's capability to initiate Hot-join request. Value After Reset: IC_SLV_HJ Exists: Always

Table 5-68 Fields for Register: HW_CAPABILITY (Continued)

Bits	Name	Memory Access	Description
17	DMA_EN	R	<p>Reflects the IC_HAS_DMA Configurable Parameter. Specifies whether controller is configured to have DMA handshaking interface.</p> <p>Value After Reset: IC_HAS_DMA Exists: Always</p>
16:11	HDR_TX_CLOCK_PERIOD	R	<p>Reflects the IC_HDR_TX_CLK_PERIOD Configurable Parameter.</p> <p>Value After Reset: IC_HDR_TX_CLK_PERIOD Exists: Always</p>
10:5	CLOCK_PERIOD	R	<p>Reflects the IC_CLK_PERIOD Configurable Parameter</p> <p>Value After Reset: IC_CLK_PERIOD Exists: Always</p>
4	HDR_TS_EN	R	<p>Reflects the IC_SPEED_HDR_TS Configurable Parameter. Specifies the Controllers capability to perform HDR-TS transfers.</p> <ul style="list-style-type: none"> ■ 0: HDR-TS not supported ■ 1: HDR-TS supported <p>Value After Reset: IC_SPEED_HDR_TS Exists: Always</p>
3	HDR_DDR_EN	R	<p>Reflects the IC_SPEED_HDR_DDR Configurable Parameter. Specifies the Controllers capability to perform HDR-DDR transfers.</p> <ul style="list-style-type: none"> ■ 0: HDR-DDR not supported ■ 1: HDR-DDR supported <p>Value After Reset: IC_SPEED_HDR_DDR Exists: Always</p>

Table 5-68 Fields for Register: HW_CAPABILITY (Continued)

Bits	Name	Memory Access	Description
2:0	DEVICE_ROLE_CONFIG	R	<p>Reflects the IC_DEVICE_ROLE Configurable Parameter. Specifies the configured role of DWC_mipi_i3c controller</p> <ul style="list-style-type: none"> ■ 1: Master Only ■ 2: Programmable Master-Slave ■ 3: Secondary Master ■ 4: Slave Only <p>Value After Reset: IC_DEVICE_ROLE Exists: Always</p>

5.2.4 COMMAND_QUEUE_PORT

- **Name:** COMMAND_QUEUE_PORT
- **Description:** COMMAND_QUEUE_PORT.

In Master mode of operation: Command Descriptor structure is used to schedule the transfers to devices on I3C bus. There are four types of commands defined

- Transfer Command
- Transfer Argument
- Short Data Argument
- Address Assignment Command

In Slave mode of operation: Command Queue Port is used to push commands which enables the controller to respond with data for a private read command from the master.

- **Size:** 32 bits
- **Offset:** 0xc
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-69 Fields for Register: COMMAND_QUEUE_PORT

Bits	Name	Memory Access	Description
31:0	COMMAND	W	32 bit command Value After Reset: 0x0 Exists: Always

5.2.5 RESPONSE_QUEUE_PORT

- **Name:** RESPONSE_QUEUE_PORT
- **Description:**

In Master mode of operation: The response status for each Command is written into the Response Queue by the controller if ROC (Response On Completion) bit is set or if transfer error occurs. The Response Queue can be read through this register.

It is expected that this register is be read whenever RESP_READY_STAT_INTR bit is set in INTR_STATUS register. Not doing so might result in execution of new commands getting stalled. A new command is executed only if there is space available in Response Queue to push the corresponding response.

In Slave mode of operation: The response status for each Command is written into the Response Queue. The Response Queue can be read through this register. It is expected that this register is read whenever RESP_READY_STAT_INTR bit is set in INTR_STATUS register. Not doing so might result in execution of new commands getting stalled. A new command is executed only if there is space available in Response Queue to push the corresponding response. Reset value of this register is not defined as it is internally mapped to a queue.

- **Size:** 32 bits
- **Offset:** 0x10
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-70 Fields for Register: RESPONSE_QUEUE_PORT

Bits	Name	Memory Access	Description
31:0	RESPONSE	R	<p>32 bit Response</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.2.6 RX_DATA_PORT

- **Name:** Receive Data Port Register
- **Description:** Receive Data Port Register

This register when read from, reads data from the RX Buffer. This has the same offset as TX_DATA_PORT to provide a single bi-directional data port for transmitting or receiving the data from the DWC_mipi_i3c.

- **Size:** 32 bits
- **Offset:** 0x14
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-71 Fields for Register: RX_DATA_PORT

Bits	Name	Memory Access	Description
31:0	RX_DATA_PORT	R	<p>Receive Data Port. The Receive data port is mapped to the Rx-Data Buffer. The Receive data is always packed in 4-byte aligned data words and stored in the Rx-Data Buffer. If the command length is not aligned to the 4-bytes, then the additional data bytes have to be ignored.</p> <p>Value After Reset: 0x0 Exists: Always</p>

5.2.7 TX_DATA_PORT

- **Name:** Transmit Data Port Register
- **Description:** Transmit Data Port Register

This register when written into, writes data to the TX Buffer. This has the same offset as RX_DATA_PORT to provide a single bi-directional data port for transmitting or receiving the data from the DWC_mipi_i3c.

- **Size:** 32 bits
- **Offset:** 0x14
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-72 Fields for Register: TX_DATA_PORT

Bits	Name	Memory Access	Description
31:0	TX_DATA_PORT	W	<p>Transmit Data Port</p> <p>The Transmit Data port is mapped to the Tx-Data Buffer.</p> <p>The transmit data should always be packed as 4-byte aligned data words and written to the Transmit Data Port register. If the Command length is not aligned to 4-bytes, then the additional bytes are ignored.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.2.8 IBI_QUEUE_DATA

- **Name:** In-Band Interrupt Queue Data Register
- **Description:** In-Band Interrupt Queue Data Register

This register is used in master mode of operation

This register when read from, reads the data from the IBI Queue.

- **Size:** 32 bits
- **Offset:** 0x18
- **Exists:** IC_DEVICE_ROLE<4



Table 5-73 Fields for Register: IBI_QUEUE_DATA

Bits	Name	Memory Access	Description
31:0	IBI_DATA	R	<p>In-Band Interrupt Data</p> <p>This register is mapped to the IBI Queue. The IBI Data is always packed in 4-byte aligned and put to the IBI Queue. If the incoming data is not aligned to the 4-bytes, then there are unused bytes in the end location of the IBI transfer.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.2.9 IBI_QUEUE_STATUS

- **Name:** In-Band Interrupt Queue Status Register
- **Description:** In-Band Interrupt Queue Status Register

This register is used in master mode of operation. It is expected that this register is read whenever INTR_STATUS.IBI_THLD_STS is set. This register when read from, returns the data from the IBI Queue and indicates how the controller responded to incoming IBI (SIR,MR and HJ).

- **Size:** 32 bits
- **Offset:** 0x18
- **Exists:** IC_DEVICE_ROLE<4



Table 5-74 Fields for Register: IBI_QUEUE_STATUS

Bits	Name	Memory Access	Description
31:28	IBI_STS	R	<p>IBI Received Status. Defines the master response for IBI received.</p> <ul style="list-style-type: none"> ■ 4'b0xxx: Responded with ACK ■ 4'b1xxx: Responded with NACK ■ Others: RESERVED <p>Value After Reset: 0x0 Exists: Always</p>
27:16			Reserved Field: Yes

Table 5-74 Fields for Register: IBI_QUEUE_STATUS (Continued)

Bits	Name	Memory Access	Description
15:8	IBI_ID	R	<p>IBI Identifier. The byte received after START which includes the address and the R/W bit.</p> <ul style="list-style-type: none"> ■ Device address and R/W bit in case of Slave Interrupt or Master Request. ■ Hot-Join ID and R/W bit in case of Hot-Join IBI. <p>Value After Reset: 0x0 Exists: Always</p>
7:0	DATA_LENGTH	R	<p>In-Band Interrupt data length. This field represents the length of data received along with the IBI, in bytes.</p> <p>Value After Reset: 0x0 Exists: Always</p>

5.2.10 QUEUE_THLD_CTRL

- **Name:** Queue Threshold Control Register
- **Description:** Queue Threshold Control Register

This register is used to program the threshold settings for the Queues in DWC_mipi_i3c.

- **Size:** 32 bits
- **Offset:** 0x1c
- **Exists:** (IC_DEVICE_ROLE<5)

IBI_STATUS_THLD	31:24
IBI_DATA_THLD	23:16
RESP_BUF_THLD	15:8
CMD_EMPTY_BUF_THLD	7:0

Table 5-75 Fields for Register: QUEUE_THLD_CTRL

Bits	Name	Memory Access	Description
31:24	IBI_STATUS_THLD	R/W	<p>In-Band Interrupt Status Threshold Value. Every In Band Interrupt received (with or without Payload) by I3C controller generates an IBI status. This field controls the number of IBI status entries (or greater) in the IBI queue that trigger the IBI_THLD_STAT interrupt.</p> <p>The valid range is 0 to IC_IBI_BUF_DEPTH-1. The software programs only valid values. A value of 0 sets the threshold for 1 entry, and a value of N sets the threshold for N+1 entries.</p> <p>NOTE: The valid value is only 0 if IBI with payload is selected in the configuration.</p> <p>Each IBI status entry can represent the complete (IBI payload byte size <= 4*IBI_DATA_THLD) IBI payload or a segment (IBI payload byte size > 4*IBI_DATA_THLD) of the IBI payload.</p> <p>Value After Reset: IC_DFLT_IBI_STS_THLD Exists: IC_DEVICE_ROLE<4</p>
23:16	IBI_DATA_THLD	R/W	<p>IBI Data Threshold Value This field represents the IBI data segment size in Dwords (4 bytes). The minimum supported segment size is 1 (4 bytes) and the maximum supported size is IC_IBI_DATA_BUF_DEPTH-1. The IBI_DATA_THLD field enables the slicing of the incoming IBI data and generate individual status and thereby promotes the cut-through operation in reading out the IBI data.</p> <p>Value After Reset: IC_DFLT_IBI_BUF_THLD Exists: (IC_DEVICE_ROLE<4) && (IC_HAS_IBI_DATA==1)</p>
15:8	RESP_BUF_THLD	R/W	<p>Response Buffer Threshold Value. Controls the number of entries (or greater) in the Response Queue that trigger the RESP_READY_STAT_INTR interrupt.</p> <p>The valid range is 0 to IC_RESP_BUF_DEPTH-1. The software programs only valid values. A value of 0 sets the threshold for 1 entry, and a value of N sets the threshold for N+1 entries.</p> <p>Value After Reset: IC_DFLT_RESP_BUF_THLD Exists: Always</p>

Table 5-75 Fields for Register: QUEUE_THLD_CTRL (Continued)

Bits	Name	Memory Access	Description
7:0	CMD_EMPTY_BUF_THLD	R/W	<p>Command Buffer Empty Threshold Value. Controls the number of empty locations (or greater) in the Command Queue that trigger CMD_QUEUE_READY_STAT interrupt. The valid range is 0 to IC_CMD_BUF_DEPTH-1. The software programs only valid values. Value of N ranging from 1 to IC_CMD_BUF_DEPTH-1 sets the threshold to N empty locations and a value of 0 sets the threshold to indicate that the queue is completely empty.</p> <p>Value After Reset: IC_DFLT_CMD_BUF_THLD Exists: Always</p>

5.2.11 DATA_BUFFER_THLD_CTRL

- **Name:** Data Buffer Threshold Control Register
- **Description:** Data Buffer Threshold Control Register

This register is used to program the threshold settings for the Data Buffers in DWC_mipi_i3c.

- **Size:** 32 bits
- **Offset:** 0x20
- **Exists:** (IC_DEVICE_ROLE<5)

Rsvd	31:27							
	RX_START_THLD	26:24						
Rsvd		23:19						
	TX_START_THLD	18:16						
Rsvd		15:11						
	RX_BUF_THLD	10:8						
Rsvd		7:3						
	TX_EMPTY_BUF_THLD	2:0						

Table 5-76 Fields for Register: DATA_BUFFER_THLD_CTRL

Bits	Name	Memory Access	Description
31:27			Reserved Field: Yes

Table 5-76 Fields for Register: DATA_BUFFER_THLD_CTRL (Continued)

Bits	Name	Memory Access	Description
26:24	RX_START_THLD	R/W	<p>Receive Start Threshold Value</p> <p>In master mode of operation when the controller is set up to initiate a read transfer, it waits until either one of the conditions are met before it initiates the read transfer on the I3C Interface.</p> <ul style="list-style-type: none"> ■ Data length (as specified in the command) number of locations are empty in the Receive FIFO. ■ Threshold number of locations (or more) are empty in the Receive FIFO. <p>In the slave mode of operation the slave controller ACK's a write request from Master only if threshold number of empty locations(or more) are available in its receive buffer.</p> <p>The supported values for RX_START_THLD are:</p> <ul style="list-style-type: none"> ■ 000 - 1 ■ 001 - 4 ■ 010 - 8 ■ 011 - 16 ■ 100 - 32 ■ 101 - 64 <p>Value After Reset: IC_DFLT_RX_START_THLD Exists: Always</p>
23:19			Reserved Field: Yes

Table 5-76 Fields for Register: DATA_BUFFER_THLD_CTRL (Continued)

Bits	Name	Memory Access	Description
18:16	TX_START_THLD	R/W	<p>Transfer Start Threshold Value</p> <p>In master mode of operation when the controller is set up to initiate a write transfer, it waits until either one of the following conditions are met before it initiates the write transfer on the I3C Interface.</p> <ul style="list-style-type: none"> ■ Data length (as specified in the command) number of locations are filled in the Transmit FIFO ■ Threshold number of entries (or more) are available in the Transmit FIFO <p>In slave mode of operation the slave controller ACK's a read request from Master only if either one of the following conditions are met:</p> <ul style="list-style-type: none"> ■ Data length (as specified in the command) number of locations are filled in the Transmit FIFO ■ Threshold number of entries (or more) are available in the Transmit FIFO <p>The supported values for TX_START_THLD are:</p> <ul style="list-style-type: none"> ■ 000: 1 ■ 001: 4 ■ 010: 8 ■ 011: 16 ■ 100: 32 ■ 101: 64 <p>Value After Reset: IC_DFLT_TX_START_THLD Exists: Always</p>
15:11			Reserved Field: Yes

Table 5-76 Fields for Register: DATA_BUFFER_THLD_CTRL (Continued)

Bits	Name	Memory Access	Description
10:8	RX_BUF_THLD	R/W	<p>Receive Buffer Threshold Value</p> <p><ct:cfc:1:IC_MASTER_MODE==1> This field controls the number of entries (or above) in the Receive FIFO that trigger the RX_THLD_STAT interrupt.</p> <p>If the programmed value is greater than the buffer depth, then threshold is set to IC_RX_BUF_DEPTH. The supported values for RX_BUF_THLD are</p> <ul style="list-style-type: none"> ■ 000: 1 ■ 001: 4 ■ 010: 8 ■ 011: 16 ■ 100: 32 ■ 101: 64 <p>Value After Reset: IC_DFLT_RX_BUF_THLD Exists: Always</p>
7:3			Reserved Field: Yes
2:0	TX_EMPTY_BUF_THLD	R/W	<p>Transmit Buffer Threshold Value</p> <p>This field controls the number of empty locations (or above) in the Transmit FIFO that trigger the TX_THLD_STAT interrupt.</p> <p>If the programmed value is greater than the buffer depth, then threshold is set to IC_TX_BUF_DEPTH. The supported values for TX_BUF_THLD are</p> <ul style="list-style-type: none"> ■ 000: 1 ■ 001: 4 ■ 010: 8 ■ 011: 16 ■ 100: 32 ■ 101: 64 <p>Value After Reset: IC_DFLT_TX_BUF_THLD Exists: Always</p>

5.2.12 IBI_QUEUE_CTRL

- **Name:** IBI Queue Control Register
- **Description:** This Register is used to control whether or not to intimate the application if an IBI request is rejected (Nacked).

This register is only used in master mode of operation

- **Size:** 32 bits
- **Offset:** 0x24
- **Exists:** IC_DEVICE_ROLE<4

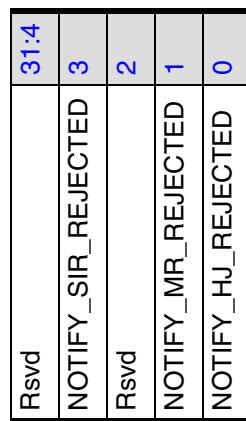


Table 5-77 Fields for Register: IBI_QUEUE_CTRL

Bits	Name	Memory Access	Description
31:4			Reserved Field: Yes

Table 5-77 Fields for Register: IBI_QUEUE_CTRL (Continued)

Bits	Name	Memory Access	Description
3	NOTIFY_SIR_REJECTED	R/W	<p>Notify Rejected Slave Interrupt Request Control. This bit is used to suppress reporting to the application about SIR request rejected.</p> <ul style="list-style-type: none"> ■ 0: Suppress passing the IBI Status to the IBI FIFO (hence not notifying the application) when a Slave Interrupt Request is NACKed and auto-disabled based on the IBI_SIR_REQ_REJECT Register. ■ 1: Writes IBI Status to the IBI FIFO (hence notifying the application) when a Slave Interrupt Request is NACKed and auto-disabled based on the IBI_SIR_REQ_REJECT Register. <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): Notify SIR Rejected Disable ■ 0x1 (ENABLED): Notify SIR Rejected Enable <p>Value After Reset: IC_DFLT_SIR_REJECTED Exists: Always</p>
2			Reserved Field: Yes
1	NOTIFY_MR_REJECTED	R/W	<p>Notify Rejected Master Request Control. This bit is used to suppress reporting to the application about Master request rejected.</p> <ul style="list-style-type: none"> ■ 0: Suppress passing the IBI Status to the IBI FIFO (hence not notifying the application) when a MR Request is NACKed and auto-disabled based on the IBI_MR_REQ_REJECT Register. ■ 1: Writes IBI Status to the IBI FIFO (hence notifying the application) when a MR Request is NACKed and auto-disabled based on the IBI_MR_REQ_REJECT Register. <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): Notify Master Request Rejected Disable ■ 0x1 (ENABLED): Notify Master Request Rejected Enable <p>Value After Reset: IC_DFLT_MR_REJECTED Exists: (IC_DEVICE_ROLE==3)</p>

Table 5-77 Fields for Register: IBI_QUEUE_CTRL (Continued)

Bits	Name	Memory Access	Description
0	NOTIFY_HJ_REJECTED	R/W	<p>Notify Rejected Hot-Join Control. This bit is used to suppress reporting to the application about Hot-Join request rejected (NACK and Auto Disable).</p> <ul style="list-style-type: none"> ■ 0: Suppress passing the IBI Status to the IBI FIFO (hence not notifying the application) when a HJ Request is NACKed and auto-disabled based on the DEVICE_CTRL.HOT_JOIN_CTRL. ■ 1: Writes IBI Status to the IBI FIFO (hence notifying the application) when a HJ Request is NACKed and auto-disabled based on the DEVICE_CTRL.HOT_JOIN_CTRL. <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (DISABLED): Notify Hot-Join Rejected Disable ■ 0x1 (ENABLED): Notify Hot-Join Rejected Enable <p>Value After Reset: IC_DFLT_HJ_REJECTED Exists: Always</p>

5.2.13 IBI_MR_REQ_REJECT

- **Name:** IBI MR Request Rejection Control Register
- **Description:** IBI Master Request Rejection Control Register.

This register is only used in secondary master mode of operation

- **Size:** 32 bits
- **Offset:** 0x2c
- **Exists:** (IC_DEVICE_ROLE==3) && (IC_HAS_IBI_DATA!=1)



Table 5-78 Fields for Register: IBI_MR_REQ_REJECT

Bits	Name	Memory Access	Description
31:0	MR_REQ_REQJECT	R/W	<p>In-band Master Request Reject.</p> <p>The control bits of this field determines if the controller ACK's incoming Master Request or NACKs and Disables it. A device specific policy can be established by appropriately programming this register.</p> <ul style="list-style-type: none"> ■ 0: ACK Master Request ■ 1: NACK and send Directed DISEC CCC to disable the interrupting slave. <p>Value After Reset: 0x0 Exists: Always</p>

5.2.14 IBI_SIR_REQ_REJECT

- **Name:** IBI SIR Request Rejection Control Register

- **Description:** IBI SIR Request Rejection Control

This register is only used in master mode of operation

- **Size:** 32 bits

- **Offset:** 0x30

- **Exists:** (IC_DEVICE_ROLE<4) && (IC_HAS_IBI_DATA!=1)



Table 5-79 Fields for Register: IBI_SIR_REQ_REJECT

Bits	Name	Memory Access	Description
31:0	SIR_REQ_REJECT	R/W	<p>In-band Slave Interrupt Request Reject The application of the DWC_mipi_i3c can decide whether to send ACK or NACK for a Slave Interrupt request received from any I3C device. A device-specific response control bit is provided to select the response option. Master ACK's/NACK's the Slave Interrupt Request based on programming of control bit, corresponding to the interrupting device.</p> <ul style="list-style-type: none"> ■ 0 - ACK the SIR Request ■ 1 - NACK and send directed auto disable CCC <p>Value After Reset: 0x0 Exists: Always</p>

5.2.15 RESET_CTRL

- **Name:** Reset Control Register
- **Description:** This Register is used for general software reset and for individual buffer reset.
- **Size:** 32 bits
- **Offset:** 0x34
- **Exists:** (IC_DEVICE_ROLE<5)

BUS_RESET	31
BUS_RESET_TYPE	30:29
Rsvd	28:6
IBI_QUEUE_RST	5
RX_FIFO_RST	4
TX_FIFO_RST	3
RESP_QUEUE_RST	2
CMD_QUEUE_RST	1
SOFT_RST	0

Table 5-80 Fields for Register: RESET_CTRL

Bits	Name	Memory Access	Description
31	BUS_RESET	R/W	<p>Bus Reset. This bit is only used in master mode of operation. Write 1'b1 to this bit to exercise Bus Reset Reset Pattern Generation based on Bus Reset Type selection. This bit is cleared automatically once the Bus Reset Pattern Generation is completed.</p> <p>Value After Reset: 0x0 Exists: IC_HAS_DEVICE_RESET_SUPPORT==1 Volatile: true</p>

Table 5-80 Fields for Register: RESET_CTRL (Continued)

Bits	Name	Memory Access	Description
30:29	BUS_RESET_TYPE	R/W	<p>Bus Reset type Type of bus reset triggered by BUS_RESET field. Values: 2'b00: EXIT: Exit Pattern 2'b11: SCL_LOW_RESET Pattern Others: Reserved</p> <p>Value After Reset: 0x0 Exists: IC_HAS_DEVICE_RESET_SUPPORT==1 Volatile: true</p>
28:6			Reserved Field: Yes
5	IBI_QUEUE_RST	R/W	<p>IBI Queue Software Reset. This bit is only used in master mode of operation. Write 1'b1 to this bit to exercise IBI Queue reset This bit is cleared automatically once the IBI Queue reset is completed.</p> <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4 Volatile: true</p>
4	RX_FIFO_RST	R/W	<p>Receive Buffer Software Reset. Write 1'b1 to this bit to exercise Receive Buffer reset. This bit is cleared automatically once the Receive buffer reset is completed.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>
3	TX_FIFO_RST	R/W	<p>Transmit Buffer Software Reset Write 1'b1 to this bit to exercise Transmit Buffer reset. This bit is cleared automatically once the Transmit Buffer reset is completed.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>

Table 5-80 Fields for Register: RESET_CTRL (Continued)

Bits	Name	Memory Access	Description
2	RESP_QUEUE_RST	R/W	<p>Response Queue Software Reset Write 1'b1 to this bit to exercise Response Queue reset. This bit is cleared automatically once the Response Queue reset is complete.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>
1	CMD_QUEUE_RST	R/W	<p>Command Queue Software Reset Write 1'b1 to this bit to exercise Command Queue reset. This bit is cleared automatically once the Command Queue reset is complete.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>
0	SOFT_RST	R/W	<p>Core Software Reset. Write 1'b1 to this bit to exercise software reset. This resets all Buffers - Receive, Transmit, Command, and Response. This bit is cleared automatically once the core reset is complete.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>

5.2.16 SLV_EVENT_STATUS

- **Name:** Slave Event Status Register
- **Description:** This register indicates the status/values of some events/controls that are relevant to slave mode of operation. These values are set by Master initiated CCCs.
- **Size:** 32 bits
- **Offset:** 0x38
- **Exists:** (IC_DEVICE_ROLE<5)

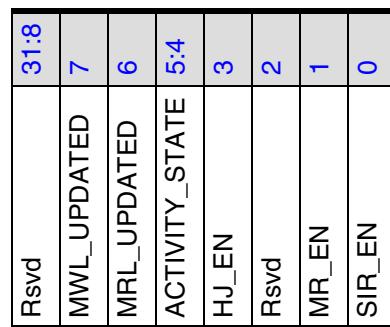


Table 5-81 Fields for Register: SLV_EVENT_STATUS

Bits	Name	Memory Access	Description
31:8			Reserved Field: Yes
7	MWL_UPDATED	R/W1C	<p>MWL Updated Status. This bit indicates a SETMWL CCC is received by the slave. The updated MWL value can be read from SLV_MAX_LEN register. This status can be cleared by writing 1'b1 to this field after reading the updated MWL.</p> <p>Value After Reset: 0x0 Exists: Always</p>
6	MRL_UPDATED	R/W1C	<p>MRL Updated Status. This bit indicates a SETMRL CCC is received by the slave. The updated MRL value can be read from SLV_MAX_LEN register. This status can be cleared by writing 1'b1 to this field after reading the updated MRL.</p> <p>Value After Reset: 0x0 Exists: Always</p>

Table 5-81 Fields for Register: SLV_EVENT_STATUS (Continued)

Bits	Name	Memory Access	Description
5:4	ACTIVITY_STATE	R	<p>Activity State Status.</p> <ul style="list-style-type: none"> ■ ENTAS0 - 00 ■ ENTAS1 - 01 ■ ENTAS2 - 10 ■ ENTAS3 - 11 <p>This bit reflects the Activity State of slave set by the Master.</p> <p>Value After Reset: 0x0 Exists: Always</p>
3	HJ_EN	R/W	<p>Hot-Join Interrupt Enable</p> <p>This bit reflects whether the Hot-Join Request Interrupts are allowed on the I3C bus or not.</p> <p>The Slave application can choose to Disable HJ Capability of the Slave Controller (if selected) by setting this field to 0 before 'Enabling' the Controller. When done so, the Slave does not initiate Hot Join and takes part in Address Assignment without initiating Hot Join. If this field is NOT set to 0 by slave application, it can be set or cleared by the I3C Master through ENEC or DISEC CCCs. Once Disabled by software, CCCs do not have any effect on this field.</p> <p>Value After Reset: IC_SLV_DFLT_HJ Exists: IC_SLV_HJ==1 Volatile: true</p>
2			Reserved Field: Yes

Table 5-81 Fields for Register: SLV_EVENT_STATUS (Continued)

Bits	Name	Memory Access	Description
1	MR_EN	R	<p>Master Request Enable. In Slave mode of operation, this bit reflects whether the controller can initiate the Master Request on the I3C bus or not. Usually, this bit is set or cleared by the I3C Master through ENEC or DISEC CCC.</p> <p>Value After Reset: 0x1 Exists: (IC_DEVICE_ROLE==3) Volatile: true</p>
0	SIR_EN	R	<p>Slave Interrupt Request Enable. In Slave mode of operation, this bit reflects whether the controller can initiate the SIR on the I3C bus or not. Usually, this bit is set or cleared by the I3C Master through ENEC or DISEC CCC.</p> <p>Value After Reset: 0x1 Exists: IC_SLV_IBI==1</p>

5.2.17 INTR STATUS

- **Name:** Interrupt Status Register
 - **Description:** Interrupt Status Register
 - **Size:** 32 bits
 - **Offset:** 0x3c
 - **Exists:** (IC_DEVICE_ROLE<5)

Rsvd	31:16
BUS_RESET_DONE_STS	15
Rsvd	14
BUSOWNER_UPDATED_STS	13
IBI_UPDATED_STS	12
READ_REQ_RECV_STS	11
DEFSLV_STS	10
TRANSFER_ERR_STS	9
DYN_ADDR_ASSGN_STS	8
Rsvd	7
CCC_UPDATED_STS	6
TRANSFER_ABORT_STS	5
RESP_READY_STS	4
CMD_QUEUE_READY_STS	3
IBI_THLD_STS	2
RX_THLD_STS	1
TX_THLD_STS	0

Table 5-82 Fields for Register: INTR_STATUS

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15	BUS_RESET_DONE_STS	R/W1C	<p>Bus Reset Pattern Generation Done Status.</p> <p>This field is used only in Master mode of operation.</p> <p>This interrupt is generated when the SCL Low Timeout Bus Reset Pattern Generation is completed. This bit can be cleared by writing 1'b1.</p> <p>Value After Reset: 0x0</p> <p>Exists: IC_HAS_DEVICE_RESET_SUPPORT==1</p> <p>Volatile: true</p>
14			Reserved Field: Yes

Table 5-82 Fields for Register: INTR_STATUS (Continued)

Bits	Name	Memory Access	Description
13	BUSOWNER_UPDATED_STS	R/W1C	<p>This interrupt is set when the role of the controller changes from being a Master to Slave or vice versa.</p> <p>This bit can be cleared by writing 1'b1.</p> <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE==3 Volatile: true</p>
12	IBI_UPDATED_STS	R/W1C	<p>IBI status is updated.</p> <p>This field is used only in slave mode of operation.</p> <p>It indicates that the IBI request initiated through SIR request register is addressed and status is updated.</p> <p>Value After Reset: 0x0 Exists: (IC_SLV_IBI==1 IC_DEVICE_ROLE==3) Volatile: true</p>
11	READ_REQ_RECV_STS	R/W1C	<p>Read Request Received.</p> <p>This field is used only in slave mode of operation.</p> <p>Read Request received from the current master when CMDQ is empty. This bit can be cleared by writing 1'b1.</p> <p>Value After Reset: 0x0 Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) Volatile: true</p>
10	DEFSLV_STS	R/W1C	<p>Define Slave CCC Received Status.</p> <p>This interrupt is generated if DEFSLV CCC is received.</p> <p>This bit can be cleared by writing 1'b1.</p> <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE==3 Volatile: true</p>
9	TRANSFER_ERR_STS	R/W1C	<p>Transfer Error Status.</p> <p>This interrupt is generated if any error occurs during transfer.</p> <p>The error type is specified in the response packet associated with the command (in ERR_STATUS field of RESPONSE_QUEUE_PORT register). This bit can be cleared by writing 1'b1.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>

Table 5-82 Fields for Register: INTR_STATUS (Continued)

Bits	Name	Memory Access	Description
8	DYN_ADDR_ASSGN_STS	R/W1C	<p>Dynamic Address Assigned Status. This field is used only in slave mode of operation. This interrupt is generated if the device's Dynamic Address is assigned through SETDASA or ENTDAA CCC. This bit can be cleared by writing 1'b1.</p> <p>Value After Reset: 0x0 Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) Volatile: true</p>
7			Reserved Field: Yes
6	CCC_UPDATED_STS	R/W1C	<p>CCC Table Updated Status. This field is used only in slave mode of operation. This interrupt is generated if any of the CCC registers are updated by I3C Master through CCC commands. This interrupt can be cleared by writing 1'b1.</p> <p>Value After Reset: 0x0 Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) Volatile: true</p>
5	TRANSFER_ABORT_STS	R/W1C	<p>Transfer Abort Status. This field is used only in master mode of operation. This interrupt is generated if transfer is aborted. This interrupt can be cleared by writing 1'b1.</p> <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4 Volatile: true</p>
4	RESP_READY_STS	R	<p>Response Queue Ready Status. This interrupt is generated when number of entries in response queue is greater than or equal to threshold value specified by RESP_BUF_THLD field in QUEUE_THLD_CTRL register. This interrupt is cleared automatically when number of entries in response buffer is less than threshold value specified.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>

Table 5-82 Fields for Register: INTR_STATUS (Continued)

Bits	Name	Memory Access	Description
3	CMD_QUEUE_READY_STS	R	<p>Command Queue Ready.</p> <p>This interrupt is generated when number of empty locations in command queue is greater than or equal to threshold value specified by CMD_EMPTY_BUF_THLD field in QUEUE_THLD_CTRL register. This interrupt is cleared automatically when number of empty locations in command buffer is less than threshold value specified.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>
2	IBI_THLD_STS	R	<p>IBI Buffer Threshold Status.</p> <p>This field is only used in master mode of operation. This interrupt is generated when number of entries in IBI buffer is greater than or equal to threshold value specified by IBI_BUF_THLD field in QUEUE_THLD_CTRL register. This interrupt is cleared automatically when number of entries in IBI buffer is less than threshold value specified.</p> <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4 Volatile: true</p>
1	RX_THLD_STS	R	<p>Receive Buffer Threshold Status.</p> <p>This interrupt is generated when number of entries in receive buffer is greater than or equal to threshold value specified by RX_BUF_THLD field in DATA_BUFFER_THLD_CTRL register. This interrupt is cleared automatically when number of entries in receive buffer is less than threshold value specified.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>

Table 5-82 Fields for Register: INTR_STATUS (Continued)

Bits	Name	Memory Access	Description
0	TX_THLD_STS	R	<p>Transmit Buffer Threshold Status This interrupt is generated when the number of empty locations in transmit buffer is greater than or equal to threshold value specified by TX_EMPTY_BUF_THLD field in DATA_BUFFER_THLD_CTRL register. This interrupt is cleared automatically when number of empty locations in transmit buffer is less than threshold value specified.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>

5.2.18 INTR_STATUS_EN

- **Name:** Interrupt Status Enable Register
- **Description:** Interrupt Status Enable Register.

The interrupt status is updated in INTR_STATUS register only if corresponding Status Enable bit is set.

- **Size:** 32 bits
- **Offset:** 0x40
- **Exists:** (IC_DEVICE_ROLE<5)

Rsvd	31:16
	BUS_RESET_DONE_STS_EN
Rsvd	15
	14
	BUSOWNER_UPDATED_STS_EN
	13
	IBI_UPDATED_STS_EN
	12
	READ_REQ_RECV_STS_EN
	11
	DEFSLV_STS_EN
	10
	TRANSFER_ERR_STS_EN
	9
	DYN_ADDR_ASSGN_STS_EN
	8
Rsvd	7
	CCC_UPDATED_STS_EN
	6
	TRANSFER_ABORT_STS_EN
	5
	RESP_READY_STS_EN
	4
	CMD_QUEUE_READY_STS_EN
	3
	IBI_THLD_STS_EN
	2
	RX_THLD_STS_EN
	1
	TX_THLD_STS_EN
	0

Table 5-83 Fields for Register: INTR_STATUS_EN

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15	BUS_RESET_DONE_STS_EN	R/W	<p>Bus Reset Pattern Generation Done Status Enable. This field is used only in Master mode of operation.</p> <p>Value After Reset: 0x0 Exists: IC_HAS_DEVICE_RESET_SUPPORT==1</p>
14			Reserved Field: Yes

Table 5-83 Fields for Register: INTR_STATUS_EN (Continued)

Bits	Name	Memory Access	Description
13	BUSOWNER_UPDATED_STS_EN	R/W	Bus owner Updated Status Enable Value After Reset: 0x0 Exists: IC_DEVICE_ROLE==3
12	IBI_UPDATED_STS_EN	R/W	IBI Updated Status Enable This field is used in slave mode of operation. Value After Reset: 0x0 Exists: (IC_SLV_IBI==1 IC_DEVICE_ROLE==3)
11	READ_REQ_RECV_STS_EN	R/W	Read Request Received Status Enable This field is used in slave mode of operation. Value After Reset: 0x0 Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)
10	DEFSLV_STS_EN	R/W	Define Slave CCC Received Status Enable Value After Reset: 0x0 Exists: IC_DEVICE_ROLE==3
9	TRANSFER_ERR_STS_EN	R/W	Transfer Error Status Enable Value After Reset: 0x0 Exists: Always
8	DYN_ADDR_ASSGN_STS_EN	R/W	Dynamic Address Assigned Status Enable This field is used in slave mode of operation. Value After Reset: 0x0 Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)
7			Reserved Field: Yes
6	CCC_UPDATED_STS_EN	R/W	CCC Table Updated Status Enable. This field is used in slave mode of operation. Value After Reset: 0x0 Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)

Table 5-83 Fields for Register: INTR_STATUS_EN (Continued)

Bits	Name	Memory Access	Description
5	TRANSFER_ABORT_STS_EN	R/W	<p>Transfer Abort Status Enable. This field is used only in master mode of operation.</p> <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4</p>
4	RESP_READY_STS_EN	R/W	<p>Response Queue Ready Status Enable</p> <p>Value After Reset: 0x0 Exists: Always</p>
3	CMD_QUEUE_READY_STS_EN	R/W	<p>Command Queue Ready Status Enable</p> <p>Value After Reset: 0x0 Exists: Always</p>
2	IBI_THLD_STS_EN	R/W	<p>IBI Buffer Threshold Status Enable. This field is used only in master mode of operation.</p> <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4</p>
1	RX_THLD_STS_EN	R/W	<p>Receive Buffer Threshold Status Enable</p> <p>Value After Reset: 0x0 Exists: Always</p>
0	TX_THLD_STS_EN	R/W	<p>Transmit Buffer Threshold Status Enable.</p> <p>Value After Reset: 0x0 Exists: Always</p>

5.2.19 INTR_SIGNAL_EN

- **Name:** Interrupt Signal Enable Register
- **Description:** Interrupt Signal Enable Register

The interrupt pin is triggered based on INTR_STATUS only if corresponding Signal Enable bit is set.

- **Size:** 32 bits
- **Offset:** 0x44
- **Exists:** (IC_DEVICE_ROLE<5)

Rsvd	31:16	
	15	
Rsvd	14	
BUSOWNER_UPDATED_SIGNAL_EN	13	
IBI_UPDATED_SIGNAL_EN	12	
READ_REQ_RECV_SIGNAL_EN	11	
DEFSLV_SIGNAL_EN	10	
TRANSFER_ERR_SIGNAL_EN	9	
DYN_ADDR_ASSGN_SIGNAL_EN	8	
Rsvd	7	
CCC_UPDATED_SIGNAL_EN	6	
TRANSFER_ABORT_SIGNAL_EN	5	
RESP_READY_SIGNAL_EN	4	
CMD_QUEUE_READY_SIGNAL_EN	3	
IBI_THLD_SIGNAL_EN	2	
RX_THLD_SIGNAL_EN	1	
TX_THLD_SIGNAL_EN	0	

Table 5-84 Fields for Register: INTR_SIGNAL_EN

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15	BUS_RESET_DONE_SIGNAL_E N	R/W	Bus Reset Pattern Generation Done Signal Enable. This field is used only in Master mode of operation. Value After Reset: 0x0 Exists: IC_HAS_DEVICE_RESET_SUPPORT==1
14			Reserved Field: Yes

Table 5-84 Fields for Register: INTR_SIGNAL_EN (Continued)

Bits	Name	Memory Access	Description
13	BUSOWNER_UPDATED_SIGNAL_EN	R/W	<p>Bus owner Updated Signal Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: IC_DEVICE_ROLE==3</p>
12	IBI_UPDATED_SIGNAL_EN	R/W	<p>IBI Updated Signal Enable This field is used in slave mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_SLV_IBI==1 IC_DEVICE_ROLE==3)</p>
11	READ_REQ_RECV_SIGNAL_EN	R/W	<p>Read Request Received Signal Enable This field is used in slave mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)</p>
10	DEFSLV_SIGNAL_EN	R/W	<p>Define Slave CCC Received Signal Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: IC_DEVICE_ROLE==3</p>
9	TRANSFER_ERR_SIGNAL_EN	R/W	<p>Transfer Error Signal Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
8	DYN_ADDR_ASSGN_SIGNAL_EN	R/W	<p>Dynamic Address Assigned Signal Enable This field is used in slave mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)</p>
7			Reserved Field: Yes
6	CCC_UPDATED_SIGNAL_EN	R/W	<p>CCC Table Updated Signal Enable This field is used in slave mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)</p>
5	TRANSFER_ABORT_SIGNAL_EN	R/W	<p>Transfer Abort Signal Enable This field is used in master mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: IC_DEVICE_ROLE<4</p>

Table 5-84 Fields for Register: INTR_SIGNAL_EN (Continued)

Bits	Name	Memory Access	Description
4	RESP_READY_SIGNAL_EN	R/W	<p>Response Queue Ready Signal Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
3	CMD_QUEUE_READY_SIGNAL_EN	R/W	<p>Command Queue Ready Signal Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
2	IBI_THLD_SIGNAL_EN	R/W	<p>IBI Buffer Threshold Signal Enable This field is used in master mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: IC_DEVICE_ROLE<4</p>
1	RX_THLD_SIGNAL_EN	R/W	<p>Receive Buffer Threshold Signal Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
0	TX_THLD_SIGNAL_EN	R/W	<p>Transmit Buffer Threshold Signal Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.2.20 INTR_FORCE

- **Name:** Interrupt Force Enable Register
- **Description:** Interrupt Force Enable Register

Individual interrupts can be forcefully triggered if corresponding Force Enable bit is set, provided the corresponding bit in the INTR_STATUS_EN register is set.

- **Size:** 32 bits
- **Offset:** 0x48
- **Exists:** (IC_DEVICE_ROLE<5)

31:16	Rsvd	BUS_RESET_DONE_FORCE_EN	15
14	Rsvd	BUSOWNER_UPDATED_FORCE_EN	13
13	IBL_UPDATED_FORCE_EN	IBL_UPDATED_FORCE_EN	12
12	READ_REQ_FORCE_EN	READ_REQ_FORCE_EN	11
11	DEFSLV_FORCE_EN	DEFSLV_FORCE_EN	10
10	TRANSFER_ERR_FORCE_EN	TRANSFER_ERR_FORCE_EN	9
9	DYN_ADDR_ASSGN_FORCE_EN	DYN_ADDR_ASSGN_FORCE_EN	8
8	Rsvd		7
7	CCC_UPDATED_FORCE_EN	CCC_UPDATED_FORCE_EN	6
6	TRANSFER_ABORT_FORCE_EN	TRANSFER_ABORT_FORCE_EN	5
5	RESP_READY_FORCE_EN	RESP_READY_FORCE_EN	4
4	CMD_QUEUE_READY_FORCE_EN	CMD_QUEUE_READY_FORCE_EN	3
3	IBL_THLD_FORCE_EN	IBL_THLD_FORCE_EN	2
2	RX_THLD_FORCE_EN	RX_THLD_FORCE_EN	1
1	TX_THLD_FORCE_EN	TX_THLD_FORCE_EN	0

Table 5-85 Fields for Register: INTR_FORCE

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15	BUS_RESET_DONE_FORCE_EN	W	Bus Reset Pattern Generation Done Force Enable. This field is used only in Master mode of operation. Value After Reset: 0x0 Exists: IC_HAS_DEVICE_RESET_SUPPORT==1
14			Reserved Field: Yes

Table 5-85 Fields for Register: INTR_FORCE (Continued)

Bits	Name	Memory Access	Description
13	BUSOWNER_UPDATED_FORCE_EN	W	<p>Bus owner Updated Force Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_DEVICE_ROLE==3)</p>
12	IBI_UPDATED_FORCE_EN	W	<p>IBI Updated Force Enable This field is used in slave mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_SLV_IBI==1 IC_DEVICE_ROLE==3)</p>
11	READ_REQ_FORCE_EN	W	<p>Read Request Received Force Enable This field is used in slave mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)</p>
10	DEFSLV_FORCE_EN	W	<p>Define Slave CCC Received Force Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_DEVICE_ROLE==3)</p>
9	TRANSFER_ERR_FORCE_EN	W	<p>Transfer Error Force Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
8	DYN_ADDR_ASSGN_FORCE_EN	W	<p>Dynamic Address Assigned Force Enable This field is used in slave mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)</p>
7			Reserved Field: Yes
6	CCC_UPDATED_FORCE_EN	W	<p>CCC Table Updated Force Enable This field is used in slave mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)</p>
5	TRANSFER_ABORT_FORCE_EN	W	<p>Transfer Abort Force Enable This field is used in master mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: IC_DEVICE_ROLE<4</p>

Table 5-85 Fields for Register: INTR_FORCE (Continued)

Bits	Name	Memory Access	Description
4	RESP_READY_FORCE_EN	W	<p>Response Queue Ready Force Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
3	CMD_QUEUE_READY_FORCE_EN	W	<p>Command Queue Ready Force Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
2	IBI_THLD_FORCE_EN	W	<p>IBI Buffer Threshold Force Enable This field is used in master mode of operation.</p> <p>Value After Reset: 0x0</p> <p>Exists: IC_DEVICE_ROLE<4</p>
1	RX_THLD_FORCE_EN	W	<p>Receive Buffer Threshold Force Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
0	TX_THLD_FORCE_EN	W	<p>Transmit Buffer Threshold Force Enable</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

5.2.21 QUEUE_STATUS_LEVEL

- **Name:** Queue Status Level Register
- **Description:** Queue Status Level Register.

This register reflects the status level of the Queues in DWC_mipi_i3c.

- **Size:** 32 bits
- **Offset:** 0x4c
- **Exists:** (IC_DEVICE_ROLE<5)

Rsvd	31:29
IBI_STS_CNT	28:24
IBI_BUF_BLR	23:16
RESP_BUF_BLR	15:8
CMD_QUEUE_EMPTY_LOC	7:0

Table 5-86 Fields for Register: QUEUE_STATUS_LEVEL

Bits	Name	Memory Access	Description
31:29			Reserved Field: Yes
28:24	IBI_STS_CNT	R	<p>IBI Buffer Status Count. When IC_HAS_IBI_DATA =0, this field is reserved and always returns 0. When IC_HAS_IBI_DATA=1, Contains the number of valid IBI Status entries in the IBI Status Buffer. This field is used in master mode of operation.</p> <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4</p>

Table 5-86 Fields for Register: QUEUE_STATUS_LEVEL (Continued)

Bits	Name	Memory Access	Description
23:16	IBI_BUF_BLR	R	<p>IBI Buffer Level Value. When IC_HAS_IBI_DATA =0, Contains the number of Valid IBI Status entries in the IBI Buffer. When IC_HAS_IBI_DATA=1, Contains the number of valid IBI Data entries in the IBI Data Buffer. This field is used in master mode of operation.</p> <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4</p>
15:8	RESP_BUF_BLR	R	<p>Response Buffer Level Value. Contains the number of valid data entries in the response Buffer.</p> <p>Value After Reset: 0x0 Exists: Always</p>
7:0	CMD_QUEUE_EMPTY_LOC	R	<p>Command Queue Empty Locations. Contains the number of empty locations in the command Buffer.</p> <p>Value After Reset: IC_CMD_BUF_DEPTH Exists: Always</p>

5.2.22 DATA_BUFFER_STATUS_LEVEL

- **Name:** Data Buffer Status Level Register
- **Description:** Data Buffer Status Level Register.

This register reflects the status level of the Buffers in DWC_mipi_i3c.

- **Size:** 32 bits
- **Offset:** 0x50
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-87 Fields for Register: DATA_BUFFER_STATUS_LEVEL

Bits	Name	Memory Access	Description
x:y	RX_BUF_BLR	R	<p>Receive Buffer Level Value. Contains the number of valid data entries in the receive Buffer.</p> <p>Value After Reset: 0x0 Exists: Always Range Variable[x]: IC_BLR_REG_BIT_WD + 7 Range Variable[y]: IC_BLR_REG_BIT_WD</p>
15:8			Reserved Field: Yes
7:0	TX_BUF_EMPTY_LOC	R	<p>Transmit Buffer Empty Level Value. Contains the number of empty locations in the transmit Buffer.</p> <p>Value After Reset: IC_TX_BUF_DEPTH Exists: Always</p>

5.2.23 PRESENT_STATE

- **Name:** Present State Register
- **Description:** The user can get status of the DWC_mipi_i3c Controller from this 32-bit read only register. This register is relevant in both master and slave mode of operation and is meant to be used to get debug information related to the controllers internal states.
- **Size:** 32 bits
- **Offset:** 0x54
- **Exists:** (IC_DEVICE_ROLE<4 && IC_HAS_HCI==0) || (((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5))&& IC_HAS_EXTND_TIMING==1)

Rsvd	31:29
	28
CMD_TID	27:24
Rsvd	23:22
CM_TFR_ST_STS	21:16
Rsvd	15:14
CM_TFR_STS	13:8
Rsvd	7:3
CURRENT_MASTER	2
SDA_LINE_SIGNAL_LEVEL	1
SCL_LINE_SIGNAL_LEVEL	0

Table 5-88 Fields for Register: PRESENT_STATE

Bits	Name	Memory Access	Description
31:29			Reserved Field: Yes
28	MASTER_IDLE	R	<p>This field reflects whether the Master Controller is in Idle state or not. This bit is set when all the Queues(Command , Response, IBI) and Buffers(Transmit and Receive) are empty along with the Master State machine is in Idle state.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (MST_NOT_IDLE): Master Controller is not in IDLE State ■ 0x1 (MST_IDLE): Master Controller is in IDLE State. <p>Value After Reset: 0x1</p> <p>Exists: Always</p> <p>Volatile: true</p>

Table 5-88 Fields for Register: PRESENT_STATE (Continued)

Bits	Name	Memory Access	Description
27:24	CMD_TID	R	<p>This field reflects the Transaction-ID of the current executing command.</p> <p>Value After Reset: 0x0 Exists: Always</p>
23:22			Reserved Field: Yes
21:16	CM_TFR_ST_STS	R	<p>Current Master Transfer State Status. Indicates the state of current transfer currently executing by the DWC_mipi_i3c controller. This is valid in Master mode only.</p> <ul style="list-style-type: none"> ■ 6'h0: IDLE (Controller is Idle state, waiting for commands from application or Slave initiated In-band Interrupt) ■ 6'h1: START Generation State. ■ 6'h2: RESTART Generation State. ■ 6'h3: STOP Generation State. ■ 6'h4: START Hold Generation for the Slave Initiated START State. ■ 6'h5: Broadcast Write Address Header(7'h7E,W) Generation State. ■ 6'h6: Broadcast Read Address Header(7'h7E,R) Generation State. ■ 6'h7: Dynamic Address Assignment State. ■ 6'h8: Slave Address Generation State. ■ 6'hB: CCC Byte Generation State. ■ 6'hC: HDR Command Generation State. ■ 6'hD: Write Data Transfer State. ■ 6'hE: Read Data Transfer State. ■ 6'hF: In-Band Interrupt(SIR) Read Data State. ■ 6'h10: In-Band Interrupt Auto-Disable State ■ 6'h11: HDR-DDR CRC Data Generation/Receive State. ■ 6'h12: Clock Extension State. ■ 6'h13: Halt State. <p>Value After Reset: 0x0 Exists: IC_DEVICE_ROLE<4</p>
15:14			Reserved Field: Yes

Table 5-88 Fields for Register: PRESENT_STATE (Continued)

Bits	Name	Memory Access	Description
13:8	CM_TFR_STS	R	<p>Transfer Type Status Indicates the type of transfer currently executing by the DWC_mipi_i3c controller.</p> <p>In Master mode of operation:</p> <ul style="list-style-type: none"> ■ 6'h0: IDLE (Controller is in Idle state, waiting for commands from application or Slave initiated In-band Interrupt) ■ 6'h1: Broadcast CCC Write Transfer. ■ 6'h2: Directed CCC Write Transfer. ■ 6'h3: Directed CCC Read Transfer. ■ 6'h4: ENTDAAS Address Assignment Transfer. ■ 6'h5: SETDASA Address Assignment Transfer. ■ 6'h6: Private I3C SDR Write Transfer. ■ 6'h7: Private I3C SDR Read Transfer. ■ 6'h8: Private I2C SDR Write Transfer. ■ 6'h9: Private I2C SDR Read Transfer. ■ 6'hA: Private HDR Ternary Symbol(TS) Write Transfer. ■ 6'hB: Private HDR Ternary Symbol(TS) Read Transfer. ■ 6'hC: Private HDR Double-Data Rate(DDR) Write Transfer. ■ 6'hD: Private HDR Double-Data Rate(DDR) Read Transfer. ■ 6'hE: Servicing In-Band Interrupt Transfer. ■ 6'hF: Halt state (Controller is in Halt State, waiting for the application to resume through DEVICE_CTRL Register) <p>In Slave mode of operation:</p> <ul style="list-style-type: none"> ■ 4'h0: IDLE (Controller is in Idle state) ■ 4'h1: Hot-Join transfer state ■ 4'h2: IBI transfer state ■ 4'h3: Master write transfer ongoing ■ 4'h4: Read data prefetch state ■ 4'h5: Master read transfer ongoing ■ 4'h6: Slave controller in Halt State waiting for resume from application <p>Value After Reset: 0x0 Exists: Always</p>

Table 5-88 Fields for Register: PRESENT_STATE (Continued)

Bits	Name	Memory Access	Description
7:3			Reserved Field: Yes
2	CURRENT_MASTER	R	<p>This Bit is used to check whether the Master is Current Master or not. The Current Master is the Master that owns the SCL line.</p> <p>If this bit is set to 0, the Master is not Current Master and requires to request and the ownership before initiating any transfer on the line.</p> <p>If this bit is set to 1, the Master is the Current Master and can initiate the transfers on the line.</p> <ul style="list-style-type: none"> ■ 0: Master is not Current Master ■ 1: Master is Current Master <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (NOT_BUS_OWNER): Master is not a Current Master ■ 0x1 (BUS_OWNER): Master is Current Master <p>Value After Reset: 0x0</p> <p>Exists: IC_DEVICE_ROLE<4</p> <p>Volatile: true</p>
1	SDA_LINE_SIGNAL_LEVEL	R	<p>This bit is used to check the SDA line level to recover from errors and for debugging. This bit reflects the value of synchronized sda_in_a signal. This is valid in Master mode only.</p> <p>Value After Reset: 0x1</p> <p>Exists: IC_DEVICE_ROLE<4</p> <p>Volatile: true</p>
0	SCL_LINE_SIGNAL_LEVEL	R	<p>This bit is used to check the SCL line level to recover from errors and for debugging. This bit reflects the value of synchronized scl_in_a signal. This is valid in Master mode only</p> <p>Value After Reset: 0x1</p> <p>Exists: IC_DEVICE_ROLE<4</p> <p>Volatile: true</p>

5.2.24 CCC DEVICE STATUS

- **Name:** Device Operating Status Register
 - **Description:** Device Operating Status Register.

This register is applicable only for I3C Slave mode of operation. This register reflects the data which the Slave controller sends in response to GETSTATUS CC by the Master.

- **Size:** 32 bits
 - **Offset:** 0x58
 - **Exists:** (((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)))

Rsvd	31:14
FRAME_ERROR	13
BUFFER_NOT_AVAIL	12
DATA_NOT_READY	11
OVERFLOW_ERR	10
SLAVE_BUSY	9
UNDERFLOW_ERR	8
ACTIVITY_MODE	7:6
PROTOCOL_ERR	5
Rsvd	4
PENDING_INTR	3:0

Table 5-89 Fields for Register: CCC_DEVICE_STATUS

Bits	Name	Memory Access	Description
31:14			Reserved Field: Yes
13	FRAME_ERROR	R	<p>Frame Error</p> <p>This bit is set when private write request from Master has frame error in HDR-DDR/HDR-TSP/TSL mode. This is cleared only after Master reads the device status through GETSTATUS_CCC.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p> <p>Volatile: true</p>

Table 5-89 Fields for Register: CCC_DEVICE_STATUS (Continued)

Bits	Name	Memory Access	Description
12	BUFFER_NOT_AVAIL	R	<p>Buffer not available</p> <p>This bit is set when private write request from Master is NACKED because of RX buffer not having RX_BUF_THLD number of empty locations or Response buffer is full. In SDR mode of operation this is cleared when the Master issues GET_STATUS CCC or upon space becoming available in the buffer and the successful completion of the next write transfer. In HDR mode of operation it is cleared only when Master issues GET_STATUS CCC.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>
11	DATA_NOT_READY	R	<p>Data not ready</p> <p>This bit is set when private read request from Master is NACKED because of any of the following conditions</p> <ul style="list-style-type: none"> ■ Command FIFO Empty. ■ Transmit FIFO threshold is not met. ■ Response FIFO Full. <p>This is cleared when the Master issues GET_STATUS CCC or upon successful completion of the subsequent read transfer.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>
10	OVERFLOW_ERR	R	<p>Overflow Error</p> <p>Overflow error condition detected during master write transfer. This is cleared only after master reads the Device Status through GETSTATUS CCC.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>

Table 5-89 Fields for Register: CCC_DEVICE_STATUS (Continued)

Bits	Name	Memory Access	Description
9	SLAVE_BUSY	R	<p>Slave Busy This bit is set if any change is made by the current master in to MRL register or occurrence of any error. It is cleared after slave application resumes the slave operation by writing 1'b1 in RESUME field of Device Control Register.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>
8	UNDERFLOW_ERR	R	<p>Underflow error</p> <p>Under Flow Error during private master read transfer. This bit is set if slave controller terminates a read transfer because of unavailability of data in the transmit buffer. This is cleared only after master reads the Device Status through GETSTATUS CCC.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>
7:6	ACTIVITY_MODE	R	<p>Activity Mode This field reflects the input port signal act_mode.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>
5	PROTOCOL_ERR	R	<p>Protocol Error</p> <p>This bit is set when the slave controller encounters a Parity/CRC error during write data transfer.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>
4			Reserved Field: Yes
3:0	PENDING_INTR	R	<p>Pending Interrupt</p> <p>This field reflects the value driven on pending_int input port.</p> <p>Value After Reset: 0x0 Exists: Always Volatile: true</p>

5.2.25 DEVICE_ADDR_TABLE_POINTER

- **Name:** Pointer for Device Address Table Registers.

- **Description:** Pointer for Device Address Table

This register is used in master mode of operation.

- **Size:** 32 bits
- **Offset:** 0x5c
- **Exists:** IC_DEVICE_ROLE<4



Table 5-90 Fields for Register: DEVICE_ADDR_TABLE_POINTER

Bits	Name	Memory Access	Description
31:16	DEV_ADDR_TABLE_DEPTH	R	Depth of Device Address Table Value After Reset: IC_DEV_ADDR_TABLE_BUF_DEPTH Exists: Always

Table 5-90 Fields for Register: DEVICE_ADDR_TABLE_POINTER (Continued)

Bits	Name	Memory Access	Description
15:0	P_DEV_ADDR_TABLE_START_A DDR	R	<p>Start Address of Device Address Table.</p> <p>Value After Reset: IC_DFLT_DEV_ADDR_TABLE_START_ADDR</p> <p>Exists: Always</p>

5.2.26 DEV_CHAR_TABLE_POINTER

- **Name:** Pointer for Device Characteristics Table
- **Description:** Pointer for Device Characteristics Table

This register is used in master mode of operation.

- **Size:** 32 bits
- **Offset:** 0x60
- **Exists:** IC_DEVICE_ROLE<4&&IC_HAS_DAA==1 && IC_HAS_HCI==0

PRESENT_DEV_CHAR_TABLE_IDX	x19
DEV_CHAR_TABLE_DEPTH	18:12
P_DEV_CHAR_TABLE_START_ADDR	11:0

Table 5-91 Fields for Register: DEV_CHAR_TABLE_POINTER

Bits	Name	Memory Access	Description
x:19	PRESENT_DEV_CHAR_TABLE_I NDX	R/W	<p>Current index of Device Characteristics Table. This field returns the current location of Device Characteristics Table index. Initially, this index points to 0. Once the complete characteristics information of a Slave device is written into Device Characteristics Table during ENTDA, this index increments by 1. The first winning device information is stored in Device Characteristics Table index 0, the second winning device information in index 1, and so on.</p> <p>If required, this index can be used to override the location, where characteristic information of Slave devices on the I3C bus are written during ENTDA. Hence, this field is useful only if the device is Current Master. During DEFSLV CCC, the index always starts from 0.</p> <p>In Non-current Master, this field is always read-only.</p> <p>Value After Reset: 0x0 Exists: (IC_HAS_DAA == 1) && (IC_RAM_DAA_ABW > 2) Range Variable[x]: IC_RAM_DAA_ABW + 16</p>
18:12	DEV_CHAR_TABLE_DEPTH	R	<p>Depth of Device Characteristics Table</p> <p>Value After Reset: IC_DEV_CHAR_TABLE_BUF_DEPTH Exists: Always</p>
11:0	P_DEV_CHAR_TABLE_START_A DDR	R	<p>Start Address of Device Characteristics Table.</p> <p>Value After Reset: IC_DFLT_DEV_CHAR_TABLE_START_ADDR Exists: Always</p>

5.2.27 VENDOR_SPECIFIC_REG_POINTER

- **Name:** Pointer for Vendor specific Registers.
 - **Description:** Pointer for Vendor Specific Registers.
- This register is used in master mode of operation.

- **Size:** 32 bits
- **Offset:** 0x6c
- **Exists:** IC_DEVICE_ROLE<4



Table 5-92 Fields for Register: VENDOR_SPECIFIC_REG_POINTER

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15:0	P_VENDOR_REG_START_ADDR	R	Start Address of Vendor specific registers. Value After Reset: 0xb0 Exists: Always

5.2.28 SLV_MIPI_ID_VALUE

- **Name:** Provisional ID Register
- **Description:** I3C MIPI Manufacturer ID Register.
This register is used in slave mode of operation.

- **Size:** 32 bits
- **Offset:** 0x70
- **Exists:** (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5) && (IC_SLV_UNIQUE_ID_PROG==1)

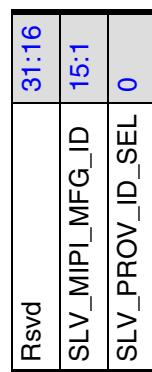


Table 5-93 Fields for Register: SLV_MIPI_ID_VALUE

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15:1	SLV_MIPI_MFG_ID	R/W	Specifies the MIPI Manufacturer ID. (PID[47:33]). Value After Reset: 0x0 Exists: Always
0	SLV_PROV_ID_SEL	R/W	Specifies the Provisional ID Type Selector (PID[32]). (1'b1: Random Value, 1'b0: Vendor Fixed Value) Value After Reset: 0x0 Exists: Always

5.2.29 SLV_PID_VALUE

- **Name:** Provisional ID Register
 - **Description:** I3C Normal Provisional ID Register.
- This register is used in slave mode of operation.

- **Size:** 32 bits
- **Offset:** 0x74
- **Exists:** (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)

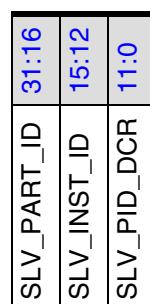


Table 5-94 Fields for Register: SLV_PID_VALUE

Bits	Name	Memory Access	Description
31:16	SLV_PART_ID	* Varies	<p>Specifies the Part ID of DWC_mipi_i3c device (PID[31:16])</p> <p>Value After Reset: "(IC_SLV_UNIQUE_ID_PROG==1) ? \"0x0\" : \"IC_SLV_PART_ID\""</p> <p>Exists: Always</p> <p>Memory Access: "(IC_SLV_UNIQUE_ID_PROG==1) ? \"read-write\" : \"read-only\""</p>
15:12	SLV_INST_ID	R/W	<p>This field is used to program the instance ID of the Slave. The reset value of this register is taken from input port 'inst_id'.</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>

Table 5-94 Fields for Register: SLV_PID_VALUE (Continued)

Bits	Name	Memory Access	Description
11:0	SLV_PID_DCR	* Varies	<p>Specifies the additional 12-bit ID of DWC_mipi_i3c device (PID[11:0]).</p> <p>Value After Reset: "(IC_SLV_UNIQUE_ID_PROG==1) ? \"0x0\" : \"IC_SLV_PID_DCR\""</p> <p>Exists: Always</p> <p>Memory Access: "(IC_SLV_UNIQUE_ID_PROG==1) ? \"read-write\" : \"read-only\""</p>

5.2.30 SLV_CHAR_CTRL

- **Name:** I3C Slave Characteristic Register
- **Description:** I3C Slave Characteristic Register.

This register is used in slave mode of operation.

- **Size:** 32 bits
- **Offset:** 0x78
- **Exists:** (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)

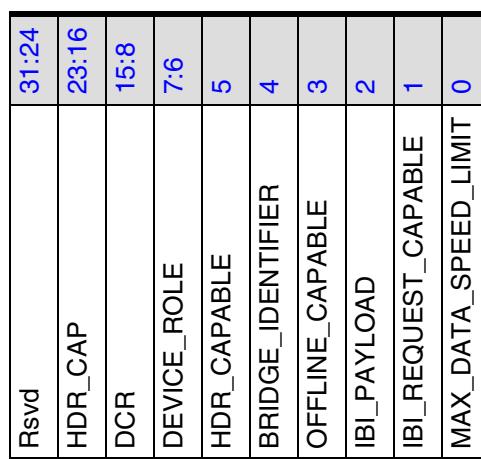


Table 5-95 Fields for Register: SLV_CHAR_CTRL

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:16	HDR_CAP	R	<p>I3C Device HDR Capability Register Value. HDR_CAP[2] - HDR Mode 2 (IC_SPEED_HDR_TS) HDR_CAP[1] - HDR Mode 1 (IC_SPEED_HDR_TS) HDR_CAP[1] - HDR Mode 0 (IC_SPEED_HDR_DDR) Others - Reserved</p> <p>Value After Reset: =::DWC_mipi_i3c::dflt_hdr_cap Exists: Always</p>

Table 5-95 Fields for Register: SLV_CHAR_CTRL (Continued)

Bits	Name	Memory Access	Description
15:8	DCR	* Varies	I3C Device Characteristic Value. Value After Reset: IC_SLV_DCR_VALUE Exists: Always Memory Access: "(IC_SLV_UNIQUE_ID_PROG==1) ? \"read-write\" : \"read-only\""
7:6	DEVICE_ROLE	R/W	Device Role field in Bus Characteristic Register (BCR[7:6]). This field is set to 1 by default if configuration parameter IC_DEVICE_ROLE is set to 3(Secondary Master). The application is not expected to change the role once configured. But if the application chooses to operate the secondary master configuration (IC_DEVICE_ROLE==3) as "Slave Only" through programming, then the Device Role can be overwritten as Slave (BCR[7:6] = 2'b00). The field is Reset to its default value upon HW/SW Reset. SW must program the values again after issuing reset. Value After Reset: "(IC_DEVICE_ROLE==3) ? \"0x1\" : \"0x0\"" Exists: Always
5	HDR_CAPABLE	R/W	SDR Only or SDR and HDR Capable field in Bus Characteristic Register (BCR[5]). This bit is set if any of the configuration parameter IC_SPEED_HDR_TS or IC_SPEED_HDR_DDR is set to 1. It is to be noted that the programming this field to 0 does not Disable the HDR Feature itself. This bit can be modified by the application if it does not want to advertize Slaves HDR capability to Master. The field is Reset to its default value upon HW/SW Reset. SW must program the values again after issuing reset. Value After Reset: IC_SLV_HDR Exists: Always
4	BRIDGE_IDENTIFIER	R	Bridge Identifier field in Bus Characteristic Register (BCR[4]). This bit is set if configuration parameter IC_SLV_BRIDGE is set to 1. Value After Reset: IC_SLV_BRIDGE Exists: Always
3	OFFLINE_CAPABLE	R	Offline Capable field in Bus Characteristic Register (BCR[3]). This bit is set if configuration parameter IC_SLV_OFFLINE_CAP is set to 1. Value After Reset: IC_SLV_OFFLINE_CAP Exists: Always

Table 5-95 Fields for Register: SLV_CHAR_CTRL (Continued)

Bits	Name	Memory Access	Description
2	IBI_PAYLOAD	R	IBI Payload field in Bus Characteristic Register (BCR[2]). This bit is set if configuration parameter IC_SLV_IBI_DATA is set to 1. Value After Reset: IC_SLV_IBI_DATA Exists: Always
1	IBI_REQUEST_CAPABLE	R	IBI Request Capable field in Bus Characteristic Register (BCR[1]). This bit is set if configuration parameter IC_SLV_IBI is set to 1. Value After Reset: IC_SLV_IBI Exists: Always
0	MAX_DATA_SPEED_LIMIT	R/W	Max Data Speed Limitation field in Bus Characteristic Register (BCR[0]). Specifies whether or not DWC_mipi_i3c has maximum data speed limitation. If this bit is set to 0, controller NACK's the GETMXDS CCC sent by Master. If this bit is set to 1, controller returns the data in MAX_DATA_SPEED and MAX_READ_TURNAROUND register in response the GETMXDS CCC sent by Master. The field is Reset to its default value upon HW/SW Reset. SW must program the values again after issuing reset. Value After Reset: IC_SLV_DATA_SPEED_LIMIT Exists: Always

5.2.31 SLV_MAX_LEN

- **Name:** I3C Max Write/Read Length Register
- **Description:** I3C Max Write/Read Length Register.

This register is used in slave mode of operation.

- **Size:** 32 bits
- **Offset:** 0x7c
- **Exists:** (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)

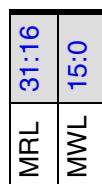


Table 5-96 Fields for Register: SLV_MAX_LEN

Bits	Name	Memory Access	Description
31:16	MRL	R	I3C Device Max Read Length. Value After Reset: IC_SLV_DFLT_MRL Exists: Always
15:0	MWL	R	I3C Device Max Write Length Value After Reset: IC_SLV_DFLT_MWL Exists: Always

5.2.32 MAX_READ_TURNAROUND

- **Name:** MXDS Maximum Read Turnaround Time Register
- **Description:** MXDS Maximum Read Turnaround Time.

This register is used in slave mode of operation and the value in this register is returned by the slave as GETMXDS CCC data.

- **Size:** 32 bits
- **Offset:** 0x80
- **Exists:** (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)



Table 5-97 Fields for Register: MAX_READ_TURNAROUND

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:0	MXDS_MAX_RD_TURN	R	<p>Specifies the maximum read turnaround time (in microseconds (us)) of DWC_mipi_i3c Slave.</p> <p>Value After Reset: IC_SLV_MXDS_MAX_RD_TURN Exists: Always</p>

5.2.33 MAX_DATA_SPEED

- **Name:** MXDS Maximum Data Speed Register
- **Description:** The values in this register are returned by the slave as GETACCMST CCC data. . .
This register is used in slave mode of operation.

- **Size:** 32 bits
- **Offset:** 0x84
- **Exists:** (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)



Table 5-98 Fields for Register: MAX_DATA_SPEED

Bits	Name	Memory Access	Description
31:19			Reserved Field: Yes

Table 5-98 Fields for Register: MAX_DATA_SPEED (Continued)

Bits	Name	Memory Access	Description
18:16	MXDS_CLK_DATA_TURN	* Varies	<p>Specifies the clock to data turnaround time (T_{sco} parameter) of DWC_mipi_i3c Slave device</p> <ul style="list-style-type: none"> ■ 0: 8ns ■ 1: 9ns ■ 2: 10ns ■ 3: 11ns ■ 4: 12ns ■ 5-7: Reserved/user Defined <p>Value After Reset: "(IC_SLV_MXDS_PROG==1) ? \"0x0\" : \"IC_SLV_MXDS_CLK_DATA_TURN\""</p> <p>Exists: Always</p> <p>Memory Access: "(IC_SLV_MXDS_PROG==1) ? \"read-write\" : \"read-only\""</p>
15:11			Reserved Field: Yes
10:8	MXDS_MAX_RD_SPEED	* Varies	<p>Specifies the Maximum Sustained Data Rate for non-CCC messages sent by DWC_mipi_i3c Slave Device to Master Device</p> <ul style="list-style-type: none"> ■ 0: 12.5MHz ■ 1: 8MHz ■ 2: 6MHz ■ 3: 4MHz ■ 4: 2MHz ■ 5-7: Reserved/User Defined <p>Value After Reset: "(IC_SLV_MXDS_PROG==1) ? \"0x0\" : \"IC_SLV_MXDS_MAX_RD_SPEED\""</p> <p>Exists: Always</p> <p>Memory Access: "(IC_SLV_MXDS_PROG==1) ? \"read-write\" : \"read-only\""</p>
7:3			Reserved Field: Yes

Table 5-98 Fields for Register: MAX_DATA_SPEED (Continued)

Bits	Name	Memory Access	Description
2:0	MXDS_MAX_WR_SPEED	* Varies	<p>Specifies the Maximum Sustained Data Rate for non-CCC messages sent by Master Device to DWC_mipi_i3c Slave device</p> <ul style="list-style-type: none"> ■ 0: 12.5MHz ■ 1: 8MHz ■ 2: 6MHz ■ 3: 4MHz ■ 4: 2MHz ■ 5-7: Reserved/User Defined <p>Value After Reset: "(IC_SLV_MXDS_PROG==1) ? \"0x0\" : \"IC_SLV_MXDS_MAX_WR_SPEED\""</p> <p>Exists: Always</p> <p>Memory Access: "(IC_SLV_MXDS_PROG==1) ? \"read-write\" : \"read-only\""</p>

5.2.34 SLV_INTR_REQ

- Name:** Slave Interrupt Request Register
- Description:** This register is used in slave mode of operation.

This register is used to program the bits that control Slave Interrupt Request and to read the status of the interrupt.

- Size:** 32 bits
- Offset:** 0x8c
- Exists:** ((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5))&& (IC_SLV_IBI==1 || IC_DEVICE_ROLE<4)

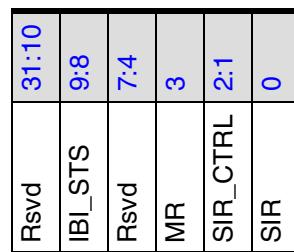


Table 5-99 Fields for Register: SLV_INTR_REQ

Bits	Name	Memory Access	Description
31:10			Reserved Field: Yes

Table 5-99 Fields for Register: SLV_INTR_REQ (Continued)

Bits	Name	Memory Access	Description
9:8	IBI_STS	R	<p>IBI Completion Status This field is common for SIR and MR</p> <ul style="list-style-type: none"> ■ 2'b00: Reserved ■ 2'b01: IBI accepted by the Master (ACK response received) ■ 2'b10: Reserved ■ 2'b11: IBI Not Attempted <p>The controller does not attempt to issue the IBI and generates the IBI Completion Status as 2'b11 if one of the following condition is true. 1. Master has not assigned the Dynamic Address. 2. Master has cleared the assigned Dynamic Address (through RSTDAA CCC). 3. Master has disabled the IBI event (through DISEC CCC). 4. The controller has switched the role to Master (applicable only for secondary master configuration).</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p>
7:4			Reserved Field: Yes
3	MR	R/W	<p>Master Request</p> <p>When set, the controller attempts to issue the MR on the I3C bus. Once issued and when the current master accepts (ACK) or if the controller is unable to issue the MR, then the controller clears this bit automatically and updates the IBI_STS field. If NACK response is received for the MR, the controller reattempts the MR upon detecting the next START condition from the master or after the Bus Available time. This bit is available only in the secondary master configuration. For other configurations, this bit is reserved and returns 0 when read. Once set, the application cannot clear this bit.</p> <p>Value After Reset: 0x0</p> <p>Exists: (IC_DEVICE_ROLE==3)</p> <p>Volatile: true</p>

Table 5-99 Fields for Register: SLV_INTR_REQ (Continued)

Bits	Name	Memory Access	Description
2:1	SIR_CTRL	R/W	<p>Slave Interrupt Request Control</p> <ul style="list-style-type: none"> ■ 2'b00: Send the Assigned Dynamic Address ■ 2'b01: Reserved ■ 2'b10: Reserved ■ 2'b11: Reserved <p>Value After Reset: 0x0 Exists: IC_SLV_IBI==1</p>
0	SIR	R/W	<p>Slave Interrupt Request</p> <p>When set, the slave controller attempts to issue the SIR on the I3C bus. Once issued and when the current master accepts (ACK) or if the controller is unable to issue the SIR, then the controller clears this bit automatically and updates the IBI_STS field. If the NACK response is received for the SIR, the controller reattempts the SIR upon detecting the next START condition from the master or after the Bus Available Time. Once set, the application cannot clear this bit.</p> <p>Value After Reset: 0x0 Exists: IC_SLV_IBI==1 Volatile: true</p>

5.2.35 SLV_TSX_SYMBL_TIMING

- **Name:** TSP/TSL Symbol Timing Register
- **Description:** TSP/TSL Symbol Timing Register

This register is used in slave mode of operation.

This Register holds the symbol time used when a device is sending data in TSP/TSL mode. The programmed value must be the desired symbol duration represented in multiples of hdr_tx_clk clock cycles.

- **Size:** 32 bits
- **Offset:** 0x90
- **Exists:** ((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5))&& IC_SPEED_HDR_TS==1



Table 5-100 Fields for Register: SLV_TSX_SYMBL_TIMING

Bits	Name	Memory Access	Description
31:6			Reserved Field: Yes
5:0	SLV_TSX_SYMBL_CNT	R/W	<p>TSP/TSL Symbol Count Value. This field is used by the controller in Slave mode to calculate the symbol duration in TSP/TSL mode while sending data. The symbol duration is count times hdr_tx_clk period.</p> <p>Value After Reset: IC_DFLT_SLV_TSX_SYMBL_CNT Exists: Always</p>

5.2.36 DEVICE_CTRL_EXTENDED

- **Name:** Device Control Extended Register
- **Description:** Device Control Extended register.

This register is relevant to both Master and Slave modes of operation and hosts functions related to Slaves disposition to incoming GETACCMST CCC from current master. It also allows for the Device Role to be fixed through Software programming in some configurations.

- **Size:** 32 bits
- **Offset:** 0xb0
- **Exists:** IC_DEVICE_ROLE<4

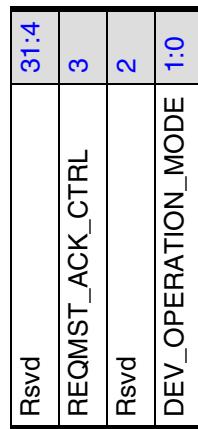


Table 5-101 Fields for Register: DEVICE_CTRL_EXTENDED

Bits	Name	Memory Access	Description
31:4			Reserved Field: Yes
3	REQMST_ACK_CTRL	R/W	<p>In Slave mode of operation, this bit serves as a control to ACK/NACK GETACCMST CCC from current master.</p> <ul style="list-style-type: none"> ■ 0: ACK GETACCMST CCC ■ 1: NACK GETACCMST CCC <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (ACK): ACK ■ 0x1 (NACK): NACK <p>Value After Reset: IC_DFLT_ACK_REQMST Exists: (IC_DEVICE_ROLE==3)</p>

Table 5-101 Fields for Register: DEVICE_CTRL_EXTENDED (Continued)

Bits	Name	Memory Access	Description
2			Reserved Field: Yes
1:0	DEV_OPERATION_MODE	R/W	<p>This bit is used to select the Device Operation Mode before the controller is enabled.</p> <p>This field is written only when the DWC_mipi_i3c is disabled.</p> <ul style="list-style-type: none"> ■ 0: Master ■ 1: Slave ■ 2: Reserved ■ 3: Reserved <p>This field is automatically updated by the controller once the role change happens in secondary master mode..</p> <p>Values:</p> <ul style="list-style-type: none"> ■ 0x0 (MASTER): Master ■ 0x1 (SLAVE): Slave <p>Value After Reset: IC_DFLT_DEV_OPERATION_MODE</p> <p>Exists: Always</p> <p>Volatile: true</p>

5.2.37 SCL_I3C_OD_TIMING

- **Name:** SCL I3C Open Drain Timing Register
- **Description:** SCL I3C Open Drain Timing Register

This register sets the SCL clock high period and low period count for I3C Open Drain transfers. The count value takes the number of core_clks to maintain the I/O SCL High/Low Period timing.

- **Size:** 32 bits
- **Offset:** 0xb4
- **Exists:** IC_DEVICE_ROLE<4 && IC_HAS_HCI==0 && IC_HAS_EXTND_TIMING==1

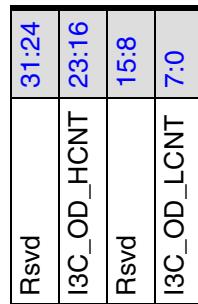


Table 5-102 Fields for Register: SCL_I3C_OD_TIMING

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:16	I3C_OD_HCNT	R/W	<p>I3C Open Drain High Count. SCL open-drain High count (I3C) for I3C transfers targeted to I3C devices.</p> <p>Value After Reset: IC_DFLT_I3C_OD_HCNT Exists: Always</p>
15:8			Reserved Field: Yes
7:0	I3C_OD_LCNT	R/W	<p>I3C Open Drain Low Count. SCL Open-drain low count for I3C transfers targeted to I3C devices.</p> <p>Value After Reset: IC_DFLT_I3C_OD_LCNT Exists: Always</p>

5.2.38 SCL_I3C_PP_TIMING

- **Name:** SCL I3C Push Pull Timing Register
- **Description:** SCL I3C Push Pull Timing Register

This register sets the SCL clock high period and low period count for I3C Push Pull transfers. The count value takes the number of core_clks to maintain the I/O SCL High/Low Period timing.

- **Size:** 32 bits
- **Offset:** 0xb8
- **Exists:** IC_DEVICE_ROLE<4 && IC_HAS_HCI==0 && IC_HAS_EXTND_TIMING==1

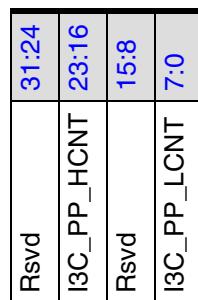


Table 5-103 Fields for Register: SCL_I3C_PP_TIMING

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:16	I3C_PP_HCNT	R/W	I3C Push Pull High Count. SCL push-pull High count for I3C transfers targeted to I3C devices. Value After Reset: IC_DFLT_I3C_PP_HCNT Exists: Always
15:8			Reserved Field: Yes
7:0	I3C_PP_LCNT	R/W	I3C Push Pull Low Count. SCL Push-pull low count for I3C transfers targeted to I3C devices. Value After Reset: IC_DFLT_I3C_PP_LCNT Exists: Always

5.2.39 SCL_I2C_FM_TIMING

- **Name:** SCL I2C Fast Mode Timing Register
- **Description:** SCL I2C Fast Mode Timing Register

This register sets the SCL clock high period and low period count for I2C Fast Mode transfers. The count value takes the number of core_clks to maintain the I/O SCL Low/High period timing.

- **Size:** 32 bits
- **Offset:** 0xbc
- **Exists:** IC_DEVICE_ROLE<4 && IC_HAS_HCI==0 && IC_HAS_EXTND_TIMING==1

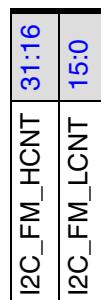


Table 5-104 Fields for Register: SCL_I2C_FM_TIMING

Bits	Name	Memory Access	Description
31:16	I2C_FM_HCNT	R/W	<p>I2C Fast Mode High Count The SCL open-drain high count timing for I2C fast mode transfers.</p> <p>Value After Reset: IC_DFLT_I2C_FM_HCNT Exists: Always</p>
15:0	I2C_FM_LCNT	R/W	<p>I2C Fast Mode Low Count The SCL open-drain low count timing for I2C fast mode transfers.</p> <p>Value After Reset: IC_DFLT_I2C_FM_LCNT Exists: Always</p>

5.2.40 SCL_I2C_FMP_TIMING

- **Name:** SCL I2C Fast Mode Plus Timing Register
- **Description:** SCL I2C Fast Mode Plus Timing Register

This register sets the SCL clock high period and low period count for I2C Fast Mode Plus transfers. The count value takes the number of core_clks to maintain the I/O SCL Low/High period timing.

- **Size:** 32 bits
- **Offset:** 0xc0
- **Exists:** IC_DEVICE_ROLE<4 && IC_HAS_HCI==0 && IC_HAS_EXTND_TIMING==1

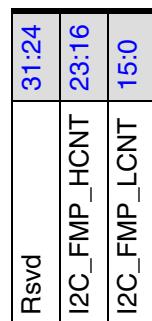


Table 5-105 Fields for Register: SCL_I2C_FMP_TIMING

Bits	Name	Memory Access	Description
31:24			Reserved Field: Yes
23:16	I2C_FMP_HCNT	R/W	<p>I2C Fast Mode Plus High Count The SCL open-drain high count timing for I2C fast mode plus transfers.</p> <p>Value After Reset: IC_DFLT_I2C_FMP_HCNT Exists: Always</p>
15:0	I2C_FMP_LCNT	R/W	<p>I2C Fast Mode Plus Low Count The SCL open-drain low count timing for I2C fast mode plus transfers.</p> <p>Value After Reset: IC_DFLT_I2C_FMP_LCNT Exists: Always</p>

5.2.41 SCL_EXT_LCNT_TIMING

- **Name:** SCL Extended Low Count Timing Register
- **Description:** SCL Extended Low Count Timing Register.

This register sets the extended low periods for the I3C transfers to allow the low data rates of the Slave devices as specified in GETMXDS CCC. The Speed field of Transfer command (COMMAND_QUEUE_PORT_TRANSFER_COMMAND) decides the selection of extended low period to achieve the lower data rate for the transfers to Slave devices.

- SDR1: Uses I3C_EXT_LCNT_1 field for the data transfer.
- SDR2: Uses I3C_EXT_LCNT_2 field for the data transfer.
- SDR3: Uses I3C_EXT_LCNT_3 field for the data transfer.
- SDR4: Uses I3C_EXT_LCNT_4 field for the data transfer.

- **Size:** 32 bits
- **Offset:** 0xc8
- **Exists:** IC_DEVICE_ROLE<4

I3C_EXT_LCNT_4	31:24
I3C_EXT_LCNT_3	23:16
I3C_EXT_LCNT_2	15:8
I3C_EXT_LCNT_1	7:0

Table 5-106 Fields for Register: SCL_EXT_LCNT_TIMING

Bits	Name	Memory Access	Description
31:24	I3C_EXT_LCNT_4	R/W	<p>I3C Extended Low Count Register 4 SDR4 uses this register field for data transfer.</p> <p>Value After Reset: IC_DFLT_EXT_LCNT_4 Exists: Always</p>

Table 5-106 Fields for Register: SCL_EXT_LCNT_TIMING (Continued)

Bits	Name	Memory Access	Description
23:16	I3C_EXT_LCNT_3	R/W	<p>I3C Extended Low Count Register 3 SDR3 uses this register field for data transfer.</p> <p>Value After Reset: IC_DFLT_EXT_LCNT_3 Exists: Always</p>
15:8	I3C_EXT_LCNT_2	R/W	<p>I3C Extended Low Count Register 2 SDR2 uses this register field for data transfer.</p> <p>Value After Reset: IC_DFLT_EXT_LCNT_2 Exists: Always</p>
7:0	I3C_EXT_LCNT_1	R/W	<p>I3C Extended Low Count Register 1 SDR1 uses this register field for data transfer.</p> <p>Value After Reset: IC_DFLT_EXT_LCNT_1 Exists: Always</p>

5.2.42 SCL_EXT_TERMN_LCNT_TIMING

- **Name:** SCL Termination Bit Low count Timing Register
- **Description:** SCL Termination Bit Low Count Timing Register

This register is used to extend the SCL Low period for Read Termination Bit.

- **Size:** 32 bits
- **Offset:** 0xcc
- **Exists:** IC_DEVICE_ROLE<4

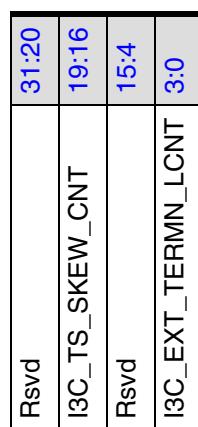


Table 5-107 Fields for Register: SCL_EXT_TERMN_LCNT_TIMING

Bits	Name	Memory Access	Description
31:20			Reserved Field: Yes
19:16	I3C_TS_SKEW_CNT	R/W	I3C HDR Ternary Skew Count. Ternary Skew Count in terms of core_clks which is used for HDR Ternary Bus Turn-Around Detection in Master Mode. Value After Reset: IC_DFLT_TS_SKEW_LCNT Exists: IC_DEVICE_ROLE<4 && IC_SPEED_HDR_TS==1
15:4			Reserved Field: Yes

Table 5-107 Fields for Register: SCL_EXT_TERMN_LCNT_TIMING (Continued)

Bits	Name	Memory Access	Description
3:0	I3C_EXT_TERMN_LCNT	R/W	<p>I3C Read Termination Bit Low count. Extended I3C Read Termination Bit low count for I3C Read transfers. Effective Termination-Bit Low Period is derived based on the SDR speed as shown below</p> <ul style="list-style-type: none"> ■ SDR0 speed: I3C_PP_LCNT + I3C_EXT_TERMN_LCNT ■ SDR1 speed: I3C_EXT_LCNT_1 + I3C_EXT_TERMN_LCNT ■ SDR2 speed: I3C_EXT_LCNT_2 + I3C_EXT_TERMN_LCNT ■ SDR3 speed: I3C_EXT_LCNT_3 + I3C_EXT_TERMN_LCNT ■ SDR4 speed: I3C_EXT_LCNT_4 + I3C_EXT_TERMN_LCNT <p>Value After Reset: IC_DFLT_TERMN_LCNT Exists: Always</p>

5.2.43 SDA_HOLD_SWITCH_DLY_TIMING

- **Name:** SDA Hold and Mode Switch Delay Timing Register
- **Description:** SDA Hold and Mode Switch Delay Timing Register

The Bits [2:0] of this register are used to shift the sda_out with respect to sda_oe while switching transfer from Open Drain timing to Push Pull timing. The bits [10:8] of this register are used to shift the sda_oe with respect to sda_out while switching transfer from Pus pull timing to Open Drain timing. The bits [18:16] of this register are used to control the hold time of SDA during transmit mode in SDR & DDR transfers.

- **Size:** 32 bits
- **Offset:** 0xd0
- **Exists:** IC_DEVICE_ROLE<4

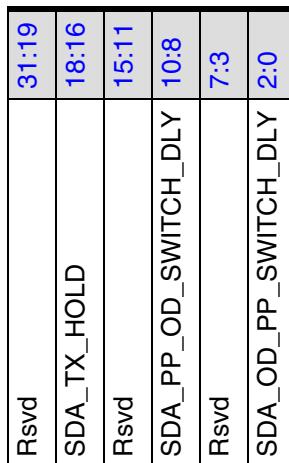


Table 5-108 Fields for Register: SDA_HOLD_SWITCH_DLY_TIMING

Bits	Name	Memory Access	Description
31:19			Reserved Field: Yes
18:16	SDA_TX_HOLD	R/W	<p>This field controls the hold time (in term of the core clock period) of the transmit data (SDA) with respect to the SCL edge in FM FM+ SDR and DDR speed mode of operations. This field is not applicable for the ternary speed modes. The valid values are 1 to 7. Others are Reserved.</p> <p>Value After Reset: IC_DFLT_SDA_TX_HOLD Exists: Always</p>

Table 5-108 Fields for Register: SDA_HOLD_SWITCH_DLY_TIMING (Continued)

Bits	Name	Memory Access	Description
15:11			Reserved Field: Yes
10:8	SDA_PP_OD_SWITCH_DLY	R/W	<p>This field is used to delay the sda_oe with respect to sda_out (in terms of the core clock period) while switching the transfer from PP1 (Push-Pull Mode SDA=1) to OD1 (Open-Drain SDA=1). The valid value can range from 0 to 4. Others are Reserved.</p> <p>Value After Reset: IC_DFLT_PP_OD_SWITCH_DLY Exists: IC_HAS_MODE_SWITCH_DLY==1</p>
7:3			Reserved Field: Yes
2:0	SDA_OD_PP_SWITCH_DLY	R/W	<p>This field is used to delay the sda_out with respect to sda_oe while switching the transfer from OD1 (Open-Drain Mode SDA=1) to PP1 (Push-Pull Mode SDA=1). The valid value can range from 0 to 4. Others are reserved.</p> <p>Value After Reset: IC_DFLT_OD_PP_SWITCH_DLY Exists: IC_HAS_MODE_SWITCH_DLY==1</p>

5.2.44 BUS_FREE_AVAIL_TIMING

- **Name:** Bus Free Timing Register
- **Description:** Bus Free and Available Timing Register

This register sets the Bus free time for initiating the transfer in master mode or generating IBI in non-current master mode.

- **Size:** 32 bits
- **Offset:** 0xd4
- **Exists:** (IC_DEVICE_ROLE<4) || (((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5))&& IC_HAS_EXTND_TIMING==1)



Table 5-109 Fields for Register: BUS_FREE_AVAIL_TIMING

Bits	Name	Memory Access	Description
31:16	BUS_AVAILABLE_TIME	R/W	<p>This register field is used only in Slave mode of operation Bus Available Count Value . This field is used by the Slave/Non-current Master to initiate an IBI after STOP condition.</p> <p>Value After Reset: IC_DFLT_BUS_AVAIL_CNT Exists: (((IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5))&& IC_HAS_EXTND_TIMING==1)</p>

Table 5-109 Fields for Register: BUS_FREE_AVAIL_TIMING (Continued)

Bits	Name	Memory Access	Description
15:0	BUS_FREE_TIME	R/W	<p>This register field is used only in Master mode of operation I3C Bus Free Count Value.</p> <p>In Pure Bus System, this field represents tCAS parameter. In Mixed Bus system, this field is expected to be programmed to tLOW of I2C Timing.</p> <p>Value After Reset: IC_DFLT_BUS_FREE_CNT Exists: IC_DEVICE_ROLE<4</p>

5.2.45 BUS_IDLE_TIMING

- **Name:** Bus Idle Timing Register
- **Description:** Bus Idle Timing Register

This register is used in slave mode of operation.

This Register programs the Bus-Idle time period used when a device dynamically joins (hot-join) the I3C bus. This register is used to store the duration, used to detect the Bus-Idle condition if SCL and SDA are held high for the mentioned duration.

- **Size:** 32 bits
- **Offset:** 0xd8
- **Exists:** (IC_DEVICE_ROLE>1) && (IC_DEVICE_ROLE<5)



Table 5-110 Fields for Register: BUS_IDLE_TIMING

Bits	Name	Memory Access	Description
31:20			Reserved Field: Yes
19:0	BUS_IDLE_TIME	R/W	<p>Bus Idle Count Value. This field is used by the controller in Slave or Non-Current Master mode to initiate Hot-Join request if the dynamic address is not valid.</p> <p>Value After Reset: IC_DFLT_BUS_IDLE_CNT Exists: Always</p>

5.2.46 SCL_LOW_MST_EXT_TIMEOUT

- **Name:** SCL_LOW_MST_TIMEOUT
- **Description:** The SCL Low Master Extended Timeout register is used to define the duration of the SCL Low Bus Reset Pattern.

- **Size:** 32 bits
- **Offset:** 0xdc
- **Exists:** IC_HAS_DEVICE_RESET_SUPPORT==1



Table 5-111 Fields for Register: SCL_LOW_MST_EXT_TIMEOUT

Bits	Name	Memory Access	Description
x:0	SCL_LOW_MST_TIMEOUT_CO UNT	R/W	<p>This count defines the number of core clock periods to count for generation of the SCL Low Bus Reset Pattern.</p> <p>Value After Reset: IC_DFLT_SCL_LOW_TIMEOUT_COUNT</p> <p>Exists: Always</p> <p>Range Variable[x]: IC_SCL_LOW_TIMEOUT_COUNT_WIDTH - 1</p>

5.2.47 I3C_VER_ID

- **Name:** DWC_mipi_i3c Version ID Register
- **Description:** This register reflects the current release number of DWC_mipi_i3c
- **Size:** 32 bits
- **Offset:** 0xe0
- **Exists:** (IC_DEVICE_ROLE<5)

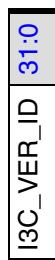


Table 5-112 Fields for Register: I3C_VER_ID

Bits	Name	Memory Access	Description
31:0	I3C_VER_ID	R	<p>Current release number</p> <p>This field indicates the Synopsys DesignWare Cores DWC_mipi_i3c current release number that is read by an application.</p> <p>For example, release number "1.00a" is represented in ASCII as 0x313030. Lower 8 bits read from this register can be ignored by the application. An application reading this register along with the I3C_VER_TYPE register, gathers details of the current release.</p> <p>Value After Reset: IC_DFLT_VER_ID Exists: Always</p>

5.2.48 I3C_VER_TYPE

- **Name:** DWC_mipi_i3c Version Type Register
- **Description:** This register reflects the current release type of DWC_mipi_i3c.

- **Size:** 32 bits
- **Offset:** 0xe4
- **Exists:** (IC_DEVICE_ROLE<5)

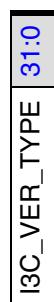


Table 5-113 Fields for Register: I3C_VER_TYPE

Bits	Name	Memory Access	Description
31:0	I3C_VER_TYPE	R	<p>Current release type</p> <p>This field indicates the Synopsys DesignWare Cores DWC_mipi_i3c current release type that is read by an application.</p> <p>For example, release type "ga" is represented in ASCII as 0x6761 and "ea" is represented as 0x6561. Lower 16 bits read from this register can be ignored by the application if release type is "ga". If release type is "ea" the lower 16 bits represents the "ea" release version.</p> <p>An application reading this register along with the I3C_VER_ID register, gathers details of the current release.</p> <p>Value After Reset: IC_DFLT_VER_TYPE</p> <p>Exists: Always</p>

5.2.49 QUEUE_SIZE_CAPABILITY

- **Name:** DWC_mipi_i3c Queue Size Capability Register
- **Description:** This register reflects the configured size of the Data Buffer and Queues in DWC_mipi_i3c.

- **Size:** 32 bits
- **Offset:** 0xe8
- **Exists:** (IC_DEVICE_ROLE<5)



Table 5-114 Fields for Register: QUEUE_SIZE_CAPABILITY

Bits	Name	Memory Access	Description
31:20			Reserved Field: Yes
19:16	IBI_BUF_SIZE	R	<p>IBI Queue Size This field reflects the configured IBI Queue size (in DWORDS) in Encoded Values.</p> <p>Values: - 0x0: 2 DWORDS - 0x1: 4 DWORDS - 0x2: 8 DWORDS - 0x3: 16 DWORDS</p> <p>Value After Reset: IC_IBI_BUF_SIZE Exists: Always</p>

Table 5-114 Fields for Register: QUEUE_SIZE_CAPABILITY (Continued)

Bits	Name	Memory Access	Description
15:12	RESP_BUF_SIZE	R	<p>Response Queue Size This field reflects the configured Response Queue size (in DWORDS) in Encoded Values.</p> <p>Values:</p> <ul style="list-style-type: none"> - 0x0: 2 DWORDS - 0x1: 4 DWORDS - 0x2: 8 DWORDS - 0x3: 16 DWORDS <p>Value After Reset: IC_RESP_BUF_SIZE Exists: Always</p>
11:8	CMD_BUF_SIZE	R	<p>Command Queue Size This field reflects the configured Command Queue size (in DWORDS) in Encoded Values.</p> <p>Values:</p> <ul style="list-style-type: none"> - 0x0: 2 DWORDS - 0x1: 4 DWORDS - 0x2: 8 DWORDS - 0x3: 16 DWORDS <p>Value After Reset: IC_CMD_BUF_SIZE Exists: Always</p>
7:4	RX_BUF_SIZE	R	<p>Receive Data Buffer Size This field reflects the configured Receive Buffer size (in DWORDS) in Encoded Values.</p> <p>Values:</p> <ul style="list-style-type: none"> - 0x0: 2 DWORDS - 0x1: 4 DWORDS - 0x2: 8 DWORDS - 0x3: 16 DWORDS - 0x4: 32 DWORDS - 0x5: 64 DWORDS <p>Value After Reset: IC_RX_DATA_BUF_SIZE Exists: Always</p>

Table 5-114 Fields for Register: QUEUE_SIZE_CAPABILITY (Continued)

Bits	Name	Memory Access	Description
3:0	TX_BUF_SIZE	R	<p>Transmit Data Buffer Size This field reflects the configured Transmit Buffer size (in DWORDS) in Encoded Values.</p> <p>Values:</p> <ul style="list-style-type: none"> - 0x0: 2 DWORDS - 0x1: 4 DWORDS - 0x2: 8 DWORDS - 0x3: 16 DWORDS - 0x4: 32 DWORDS - 0x5: 64 DWORDS <p>Value After Reset: IC_TX_DATA_BUF_SIZE Exists: Always</p>

5.2.50 DEV_CHAR_TABLE1_LOC1

- **Description:** Device Characteristic Table Location-1 of Device1 This register is used in master mode of operation. 1
- **Size:** 32 bits
- **Offset:** 0x200
- **Exists:** $4 \leq \text{IC_DEV_CHAR_TABLE_BUF_DEPTH} \&\& \text{IC_DEVICE_ROLE} < 4 \&\& \text{IC_HAS_HCI} == 0$



Table 5-115 Fields for Register: DEV_CHAR_TABLE1_LOC1

Bits	Name	Memory Access	Description
31:0	LSB_PROVISIONAL_ID	R	<p>The LSB 32-bit value of Provisional-ID</p> <p>Value After Reset: 0x0</p> <p>Exists: Always</p> <p>Testable: untestable</p> <p>Volatile: true</p>

5.2.51 SEC_DEV_CHAR_TABLE1

- **Description:** Secondary Master Device Characteristic Table Location of Device1 1
- **Size:** 32 bits
- **Offset:** 0x200
- **Exists:** $4 \leq \text{IC_DEV_CHAR_TABLE_BUF_DEPTH} \&& \text{IC_DEVICE_ROLE} == 3$

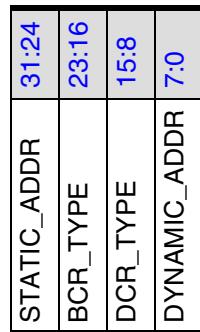


Table 5-116 Fields for Register: SEC_DEV_CHAR_TABLE1

Bits	Name	Memory Access	Description
31:24	STATIC_ADDR	R	<p>The Static Addr of Device1 Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true</p>
23:16	BCR_TYPE	R	<p>The BCR TYPE of Device1 Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true</p>
15:8	DCR_TYPE	R	<p>The DCR TYPE of Device1 Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true</p>
7:0	DYNAMIC_ADDR	R	<p>The Dynamic Addr of Device1 Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true</p>

5.2.52 DEV_CHAR_TABLE1_LOC2

- Description:** Device Characteristic Table Location-2 of Device1 This register is used in master mode of operation.
- Size:** 32 bits
- Offset:** 0x204
- Exists:** $4 \leq \text{IC_DEV_CHAR_TABLE_BUF_DEPTH} \&\& \text{IC_DEVICE_ROLE} < 4 \&\& \text{IC_HAS_HCI} == 0$



Table 5-117 Fields for Register: DEV_CHAR_TABLE1_LOC2

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15:0	MSB_PROVISIONAL_ID	R	The MSB 16-bit value of Provisional-ID Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true

5.2.53 DEV_CHAR_TABLE1_LOC3

- Description:** Device Characteristic Table Location-3 of Device1 This register is used in master mode of operation. 1
- Size:** 32 bits
- Offset:** 0x208
- Exists:** $4 \leq \text{IC_DEV_CHAR_TABLE_BUF_DEPTH} \&\& \text{IC_DEVICE_ROLE}<4 \&\& \text{IC_HAS_HCI}==0$

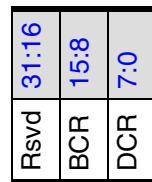


Table 5-118 Fields for Register: DEV_CHAR_TABLE1_LOC3

Bits	Name	Memory Access	Description
31:16			Reserved Field: Yes
15:8	BCR	R	Bus Characteristic Value Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true
7:0	DCR	R	Device Characteristic Value Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true

5.2.54 DEV_CHAR_TABLE1_LOC4

- **Description:** Device Characteristic Table Location-4 of Device1 This register is used in master mode of operation. 1
- **Size:** 32 bits
- **Offset:** 0x20c
- **Exists:** $4 \leq \text{IC_DEV_CHAR_TABLE_BUF_DEPTH} \&\& \text{IC_DEVICE_ROLE}<4 \&\& \text{IC_HAS_HCI}==0$



Table 5-119 Fields for Register: DEV_CHAR_TABLE1_LOC4

Bits	Name	Memory Access	Description
31:8			Reserved Field: Yes
7:0	DEV_DYNAMIC_ADDR	R	Device Dynamic Address assigned. Value After Reset: 0x0 Exists: Always Testable: untestable Volatile: true

5.2.55 DEV_ADDR_TABLE1_LOC1

- **Name:** Device Address Table Location of Device1
- **Description:** Device Address Table Location of Device1 1
- **Size:** 32 bits
- **Offset:** 0x220
- **Exists:** $1 \leq IC_DEV_ADDR_TABLE_BUF_DEPTH \&& IC_DEVICE_ROLE<4 \&& IC_HAS_HCI==0 \&& IC_HAS_DAT==1 \&& IC_HAS_IBI_DATA==1$

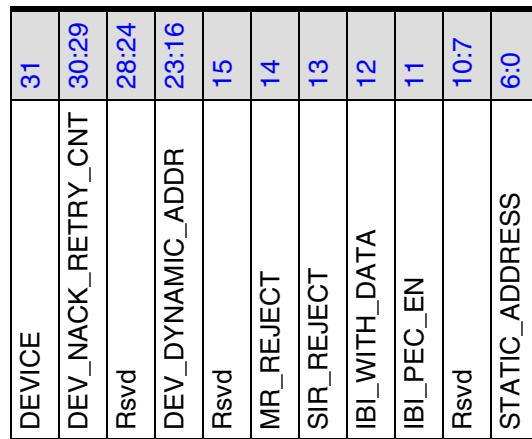


Table 5-120 Fields for Register: DEV_ADDR_TABLE1_LOC1

Bits	Name	Memory Access	Description
31	DEVICE	R/W	<p>Type of device</p> <ul style="list-style-type: none"> ■ 0: I3C ■ 1: I2C <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>

Table 5-120 Fields for Register: DEV_ADDR_TABLE1_LOC1 (Continued)

Bits	Name	Memory Access	Description
30:29	DEV_NACK_RETRY_CNT	R/W	<p>This field is used to set the Device NACK Retry count for the particular device.</p> <p>If the Device NACK's for the device address, the controller automatically retries the same device until this count expires. If the Slave does not ACK for the mentioned number of retries, then Controller generates an error response and move to the Halt state.</p> <p>This feature is used for Retry Model for the following features mentioned in the I3C Specification:</p> <ul style="list-style-type: none"> ■ Retry Model for Direct GET CCC Commands. ■ The incoming SIR-IBI matches with the slave address initiated by the Master. <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
28:24			Reserved Field: Yes
23:16	DEV_DYNAMIC_ADDR	R/W	<p>Device Dynamic Address with parity. This field consists of Dynamic address and Parity Bit. The LSB bits [22:16] should consist of Dynamic Address field indicates the address to be assigned for the winning I3C device when using ENTDA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic address used for ENTDA address assignment (~XOR(DEV_DYNAMIC_ADDR[22:16]))</p> <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
15			Reserved Field: Yes
14	MR_REJECT	R/W	<p>In-Band Master Request Reject field is used to control, per device, whether to accept Master request from Devices.</p> <ul style="list-style-type: none"> ■ 0x0 - Accept: ACK the Master Request ■ 0x1 - Reject: NACK the Master Request and send auto disable CCC. <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>

Table 5-120 Fields for Register: DEV_ADDR_TABLE1_LOC1 (Continued)

Bits	Name	Memory Access	Description
13	SIR_REJECT	R/W	<p>In-Band Slave Interrupt Request Reject field is used to control, per device, whether to accept Slave Interrupt request from Devices.</p> <ul style="list-style-type: none"> ■ 0x0 - Accept: ACK the SIR ■ 0x1 - Reject: NACK the SIR and send auto disable CCC. <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
12	IBI_WITH_DATA	R/W	<p>Mandatory one or more data bytes follow the accepted IBI from the device. Data byte continuation is indicated by T-Bit.</p> <ul style="list-style-type: none"> ■ 0x0 - IBI Without Mandatory Byte ■ 0x1 - IBI with one or more Mandatory Bytes. <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
11	IBI_PEC_EN	R/W	<p>Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether PEC check should be performed for IBI data from device. This bit also controls whether PEC byte has to be send for auto disable CCC when controller NACKs the IBI.</p> <ul style="list-style-type: none"> ■ 0x0 - Packet Error Check disabled for IBI ■ 0x1 - Packet Error Check enabled for IBI <p>This field is applicable only if configuration parameter 'IC_HAS_PEC' is set to 1.</p> <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
10:7			Reserved Field: Yes
6:0	STATIC_ADDRESS	R/W	<p>Device Static Address.</p> <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>

5.2.56 DEV_ADDR_TABLE_LOC1

- **Name:** Device Address Table of Device1
- **Description:** Device Address Table of Device1 This register is used in master mode of operation. 1
- **Size:** 32 bits
- **Offset:** 0x220
- **Exists:** 0 < IC_DEV_ADDR_TABLE_BUF_DEPTH && IC_DEVICE_ROLE<4 && IC_HAS_HCI==0 && IC_HAS_IBI_DATA==0

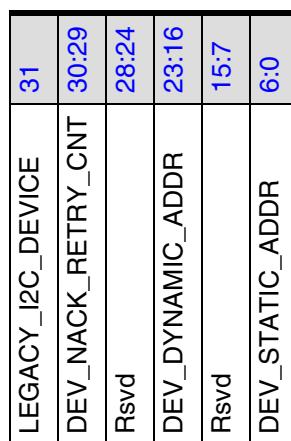


Table 5-121 Fields for Register: DEV_ADDR_TABLE_LOC1

Bits	Name	Memory Access	Description
31	LEGACY_I2C_DEVICE	R/W	<p>Legacy I2C device or not. This bit should be set to 1 if the device is a legacy I2C device.</p> <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>

Table 5-121 Fields for Register: DEV_ADDR_TABLE_LOC1 (Continued)

Bits	Name	Memory Access	Description
30:29	DEV_NACK_RETRY_CNT	R/W	<p>This field is used to set the Device NACK Retry count for the particular device.</p> <p>If the Device NACK's for the device address, the controller automatically retries the same device until this count expires. If the Slave does not ACK for the mentioned number of retries, then Controller generates an error response and move to the Halt state.</p> <p>This feature is used for Retry Model for the following features mentioned in the I3C Specification:</p> <ul style="list-style-type: none"> ■ Retry Model for Direct GET CCC Commands. ■ The incoming SIR-IBI matches with the slave address initiated by the Master. <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
28:24			Reserved Field: Yes
23:16	DEV_DYNAMIC_ADDR	R/W	<p>Device Dynamic Address with parity. This field consists of Dynamic address and Parity Bit. The LSB bits [22:16] should consist of Dynamic Address field indicates the address to be assigned for the winning I3C device when using ENTDA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic address used for ENTDA address assignment ($\sim\text{XOR}(\text{DEV_DYNAMIC_ADDR}[22:16])$)</p> <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>
15:7			Reserved Field: Yes
6:0	DEV_STATIC_ADDR	R/W	<p>Device Static Address.</p> <p>Value After Reset: 0x0 Exists: Always Testable: untestable</p>

A

Area and Power

This appendix provides reference area (in terms of gate count) and power of DWC_mipi_i3c in various modes of operation.

A.1 Area and Power

Table A-1 provides the area (in terms of gate count) and power for the various configurations.



Industry Standard 7nm synthesis library is used to extract the following area and power information.

Table A-1 Gate Count

Configuration	Configuration Type	Selected Features	Area (Unit Area: 0.05472)	Gate Count (Unit = K Gates)	Dynamic Power (Unit = MW)	Static Power (Unit = MW)	SpyGlass (DFT Coverage)
Master-Only (IC_DEVICE_ROLE==1)	Minimum	<ul style="list-style-type: none"> ■ Transfer Modes - SDR Speed ■ Clocking - Synchronous ■ Buffer Depths -Minimum 	616	11260	0.287	3e-05	99.70%
	Maximum	<ul style="list-style-type: none"> ■ Transfer Modes - SDR+DDR+TSP/TSL Speed ■ Clocking - Asynchronous ■ Buffer Depths -Maximum Depth Selected ■ DMA Handshake Interface -Enabled 	1380	25214	0.516	7e-05	99.80%
Slave-Only (IC_DEVICE_ROLE==4)	Minimum	<ul style="list-style-type: none"> ■ Transfer Modes - SDR Speed ■ Clocking - Synchronous ■ Buffer Depths -Minimum 	660	12062	0.254	3e-05	99.40%
	Maximum	<ul style="list-style-type: none"> ■ Transfer Modes - SDR+DDR+TSP/TSL Speed ■ Clocking - Asynchronous ■ Buffer Depths -Maximum Depth Selected ■ DMA Handshake Interface -Enabled ■ HJ Feature- Enabled 	1312	23976	0.239	7e-05	99.30%

Table A-1 Gate Count (Continued)

Configuration	Configuration Type	Selected Features	Area (Unit Area: 0.05472)	Gate Count (Unit = K Gates)	Dynamic Power (Unit = MW)	Static Power (Unit = MW)	SpyGlass (DFT Coverage)
Secondary-Master (IC_DEVICE_ROLE==3)	Minimum	<ul style="list-style-type: none"> ■ Transfer Modes - SDR Speed ■ Clocking - Synchronous ■ Buffer Depths -Minimum 	1238	22628	0.53	6e-05	99.60%
	Minimum	<ul style="list-style-type: none"> ■ Transfer Modes - SDR Speed ■ Clocking - Asynchronous ■ Buffer Depths -Minimum 	1570	28700	0.307	8e-05	99.70%
	Maximum	<ul style="list-style-type: none"> ■ Transfer Modes - SDR+DDR+TSP/TSL Speed ■ Clocking - Asynchronous ■ Buffer Depths -Maximum Depth Selected ■ DMA Handshake Interface -Enabled ■ HJ Feature- Enabled 	2422	44262	1.67	0.0001	99.60%
Slave-Lite (IC_DEVICE_ROLE==5)	Minimum	<ul style="list-style-type: none"> ■ Transfer Modes - SDR Speed ■ App Data Interface - 8 bit 	135	2474	0.003	7e-06	99.20%
	Maximum	<ul style="list-style-type: none"> ■ Transfer Modes - SDR+DDR Speed ■ App Data Interface - 32 bit ■ HJ Feature- Enabled 	231	4222	0.004	1e-05	99.60%

B

Synchronizer Methods

This appendix describes the synchronizer methods (blocks of synchronizer functionality) that are used in DWC_mipi_i3c to cross clock boundaries.

This appendix contains the following sections:

- “[Synchronizers Used in DWC_mipi_i3c](#)” on page [432](#)

B.1 Synchronizers Used in DWC_mipi_i3c

Each of the synchronizers and synchronizer sub-modules are comprised of verified DesignWare Basic Core (BCM) RTL designs. The BCM synchronizer designs are identified by the synchronizer type. The corresponding RTL files comprising the BCM synchronizers used in the DWC_mipi_i3c are listed in Table B-1. For more information, refer "Clock Domain Crossing" chapter in DWC_mipi_i3c Controller User Guide.

Note that certain BCM modules are contained in other BCM modules, as they are used in a building block fashion.

Table B-1 Synchronizers Used in DWC_mipi_i3c

Synchronizer Module File	Sub-module File	Synchronizer Type
DWC_mipi_i3c_bcm21.v		Simple Multiple Register Synchronizer
DWC_mipi_i3c_bcm22.v	DWC_mipi_i3c_bcm21.v	Dual Clock Pulse Synchronizer
DWC_mipi_i3c_bcm23.v	DWC_mipi_i3c_bcm21.v	Pulse Synchronizer with Acknowledge
DWC_mipi_i3c_bcm24.v		Gray Coded Synchronizer
DWC_mipi_i3c_bcm25.v	DWC_mipi_i3c_bcm21.v DWC_mipi_i3c_bcm23.v	Data Bus Synchronizer with Acknowledge
DWC_mipi_i3c_bcm37.v	DWC_mipi_i3c_bcm21.v DWC_mipi_i3c_bcm22.v	Reset Sequence Synchronizer
DWC_mipi_i3c_bcm41.v	DWC_mipi_i3c_bcm21.v	Simple Multiple Register Synchronizer with Configurable Polarity Reset.

C

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 C-1 Internal Parameters

Parameter Name	Equals To
IC_BLR_REG_BIT_WD	=((IC_HAS_EXTND_TIMING ==1) ? 16 : 8)
IC_CMD_BUF_DEPTH	=(((IC_BUF_LVL_SEL ==1) ? 4 : ((IC_BUF_LVL_SEL ==2) ? 8 : 16)))
IC_CMD_BUF_SIZE	=[(::DWC_mipi_i3c::calc_tx_data_buf_size IC_CMD_BUF_DEPTH)]
IC_DBG_DW	=(IC_DEVICE_ROLE < 5 ? 48 : 18)
IC_DEVICE_OPERATION_MODE	3
IC_DFLT_ACK_REQMST	0
IC_DFLT_BUS_TIMING_CAP_ID	8'hC0
IC_DFLT_BUS_TIMING_CAP_LEN	16'h0C
IC_DFLT_DEBUG_CAP_ID	8'h0C
IC_DFLT_DEBUG_CAP_LEN	16'h05
IC_DFLT_DEV_ADDR_TABLE_START_ADDR	([::DWC_mipi_i3c::default_dev_start_addr IC_DFLT_DEV_CHAR_TABLE_START_ADDR IC_DEV_ADDR_TABLE_BUF_DEPTH IC_DEV_CHAR_TABLE_BUF_DEPTH])

Table C-1 Internal Parameters (Continued)

Parameter Name	Equals To
IC_DFLT_DEV_CHAR_TABLE_START_ADDR	0x200
IC_DFLT_DEV_OPERATION_MODE	([::DWC_mipi_i3c::dflt_dev_op_mode])
IC_DFLT_HW_CAP_ID	8'h01
IC_DFLT_HW_CAP_LEN	16'h04
IC_DFLT_MASTER_EXT_CAP_ID	8'h02
IC_DFLT_MASTER_EXT_CAP_LEN	16'd16
IC_DFLT_OD_PP_SWITCH_DLY	3'h0
IC_DFLT_PP_OD_SWITCH_DLY	3'h0
IC_DFLT_VER_ID	32'h3130302a
IC_DFLT_VER_TYPE	32'h6C633033
IC_DMA_CORE_CLK_ASYNC	=(IC_DMA_CORE_CLK_TYPE == 1 ? 1 : 0)
IC_FW_RAM_RETIMING_EN	=((IC_FW_RAM_RETIMING ==1) ? 1 : 0)
IC_HAS_DAA	=(IC_DEVICE_ROLE<4 ? 1:0)
IC_HAS_DAT	1
IC_HAS_DAT_EN	=(IC_HAS_DAT ==1 ? 1:0)
IC_HAS_DEBUG_PORTS	1
IC_HAS_EXTND_TIMING	1
IC_HAS_MODE_SWITCH_DLY	=((IC_HAS_HCI ==1) ? 1 : 0)
IC_IBI_BUF_DEPTH	=(((IC_IBI_BUF_LVL_SEL ==1) ? 4 : ((IC_IBI_BUF_LVL_SEL ==2) ? 8 : 16)))
IC_IBI_BUF_SIZE	=[([::DWC_mipi_i3c::calc_tx_data_buf_size IC_IBI_BUF_DEPTH])]
IC_IBI_DATA_BUF_DEPTH	=((IC_DEVICE_ROLE ==1) ? ((IC_IBI_BUF_LVL_SEL ==1) ? 16 : ((IC_IBI_BUF_LVL_SEL ==2) ? 32 : 64)) : 2)
IC_INTR_IO	1
IC_INTR_POL	1
IC_MAIN_MASTER_EN	=((IC_DEVICE_ROLE < 4) ? 1 : 0)
IC_MASTER_MODE	=((IC_DEVICE_ROLE<4) ? 1 : 0)
IC_OPEN_DRAIN_CLASS_PULLUP	1

Table C-1 Internal Parameters (Continued)

Parameter Name	Equals To
IC_OPEN_DRAIN_CLASS_PULLUP_EN	$=(\text{IC_OPEN_DRAIN_CLASS_PULLUP} == 1 ? 1 : 0)$
IC_RAM_DAA_ABW	$=[::\text{DWC_mipi_i3c::plugin_log2} \text{IC_DEV_CHAR_TABLE_BUF_DEPTH}]$
IC_REGIF_MODE	0
IC_REGIF_MODE_EN	$=(\text{IC_REGIF_MODE} == 1 ? 1 : 0)$
IC_RESP_BUF_DEPTH	$=((\text{IC_BUF_LVL_SEL} == 1) ? 2 : ((\text{IC_BUF_LVL_SEL} == 2) ? 4 : 8)))$
IC_RESP_BUF_SIZE	$=[::\text{DWC_mipi_i3c::calc_tx_data_buf_size} \text{IC_RESP_BUF_DEPTH}]$
IC_RX_BUF_DEPTH	$=((\text{IC_BUF_LVL_SEL} == 1) ? 16 : ((\text{IC_BUF_LVL_SEL} == 2) ? 32 : 64)))$
IC_RX_DATA_BUF_SIZE	$=[::\text{DWC_mipi_i3c::calc_rx_data_buf_size} \text{IC_RX_BUF_DEPTH}]$
IC_SCL_LOW_TIMEOUT_COUNT_WIDTH	26
IC_SDA_SAMPLING_IN_SCL	1
IC_SLAVE_APB_EN	$=((\text{IC_DEVICE_ROLE} > 1) \&& (\text{IC_DEVICE_ROLE} < 5)) ? 1 : 0)$
IC_SLAVE_LITE_ONLY_EN	$=(\text{IC_DEVICE_ROLE} == 5) ? 1 : 0)$
IC_SLV_DCR_VALUE	8'h0
IC_SLV_DFLT_HJ	$=((\text{IC_SLV_HJ} == 1) ? 1 : 0)$
IC_SLV_HDR	$=((\text{IC_SPEED_HDR_TS} == 1 \text{IC_SPEED_HDR_DDR} == 1) ? 1 : 0)$
IC_SLV_HJ	$=((\text{IC_SLV_IBI} == 1) ? 1 : 0)$
IC_SLVIF_ADDR_WIDTH	32
IC_SLVIF_DATA_WIDTH	32
IC_SLVIF_MODE	1
IC_SLV_MIPI_MFG_ID	15'h0
IC_SLV_PART_ID	16'h0
IC_SLV_PID_DCR	12'h0
IC_SLV_PROV_ID_SEL	0

Table C-1 Internal Parameters (Continued)

Parameter Name	Equals To
IC_SLV_UNIQUE_ID_PROG	$=([::DWC_mipi_i3c::calc_dflt_uniq_id_prog IC_DEVICE_ROLE IC_SLV_MIPI_MFG_ID IC_SLV_PROVIDED_ID SEL IC_SLV_PART_ID IC_SLV_PID_DCR IC_SLV_DCR_VALUE])$
IC_SPEED_HDR_EN	$=((IC_SPEED_HDR_TS == 1 \parallel IC_SPEED_HDR_DDR == 1) ? 1 : 0)$
IC_SPEED_HDR_TS_EN	$=(IC_SPEED_HDR_TS == 1 ? 1 : 0)$
IC_SUPPORT_4K_MEM	0
IC_THLD_BIT_WD	$=((IC_SUPPORT_4K_MEM == 1) ? 4 : 3)$
IC_TX_BUF_DEPTH	$=(((IC_BUF_LVL_SEL == 1) ? 16 : ((IC_BUF_LVL_SEL == 2) ? 32 : 64)))$
IC_TX_DATA_BUF_SIZE	$=([::DWC_mipi_i3c::calc_tx_data_buf_size IC_TX_BUF_DEPTH])$
IDLE	3'd0