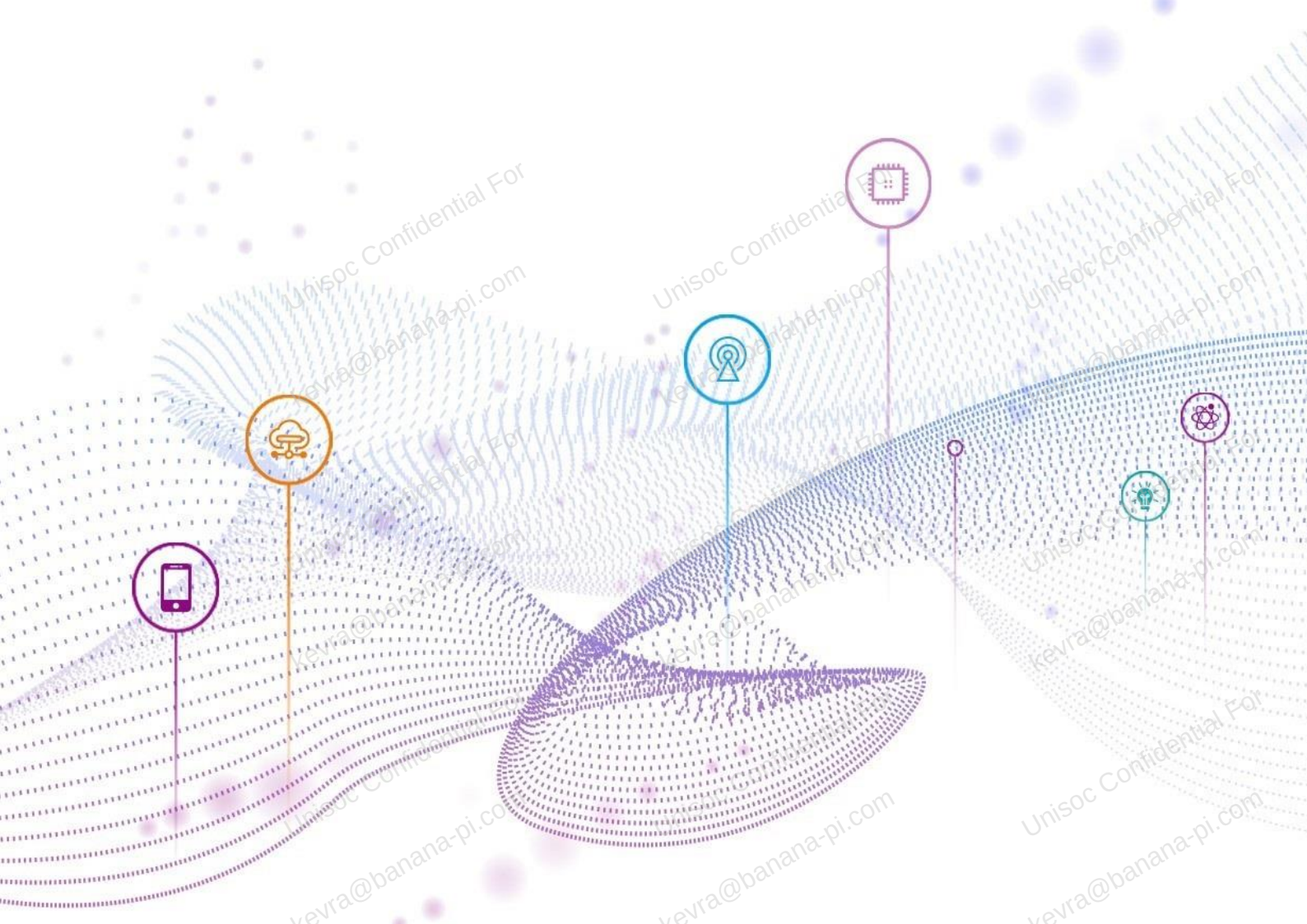




UIS7885 Device Specification

Issue
Date

V1.8
13-Jan-2026

Copyright © UNISOC (Shanghai) Technologies Co., Ltd. All rights reserved.

All data and information contained in or disclosed by this document is confidential and proprietary information of UNISOC (Shanghai) Technologies Co., Ltd. (hereafter referred as UNISOC) and all rights therein are expressly reserved. This document is provided for reference purpose, no license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, and no express and implied warranties, including but without limitation, the implied warranties of fitness for any particular purpose, and non-infringement, as well as any performance. By accepting this material, the recipient agrees that the material and the information contained therein is to be held in confidence and in trust and will not be used, copied, reproduced in whole or in part, nor its contents revealed in any manner to others without the express written permission of UNISOC. UNISOC may make any changes at any time without prior notice. Although every reasonable effort is made to present current and accurate information, UNISOC makes no guarantees of any kind with respect to the matters addressed in this document. In no event shall UNISOC be responsible or liable, directly or indirectly, for any damage or loss caused or alleged to be caused by or in connection with the use of or reliance on any such content.

Please refer to the UNISOC Documents in the UNISOC Deliverables for the use of the Deliverables. Any loss caused by the modification, customization or use of the UNISOC Deliverables in violation of the instructions in the UNISOC Documents shall be undertaken by those who conduct so. The performance indicators, test results and parameters in the UNISOC Deliverables are all obtained in the internal development and test system of UNISOC and are only for the reference. Before using UNISOC Deliverables commercially or conducting mass production of the Deliverables, comprehensive testing and debugging in combination with its own software and hardware test environment are pre-requisite.

Trademarks

UNISOC, , Spreadtrum, SPRD, RDA, and other UNISOC trademarks are trademarks of UNISOC (Shanghai) Technologies Co., Ltd. and/or its affiliated companies.

The **Bluetooth**® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by UNISOC (Shanghai) Technologies Co., Ltd. is under license.

All other trademarks and trade names mentioned in this document are the property of their respective owners.

Disclaimer

This document may include third-party content, including but not limited to third-party information, software, components, and data. UNISOC does not control or assume responsibility for any third-party content, including but not limited to its accuracy, compatibility, reliability, availability, legitimacy, appropriateness, performance, non-infringement, and status update, unless otherwise specified in this document. UNISOC does not provide any guarantee or authorization for the third-party content mentioned or referenced in this document.

User should check the availability of third-party content. If a third-party license is needed, user shall obtain it in an authorized or legal way, unless otherwise specified in this document.

UNISOC (Shanghai) Technologies Co., Ltd.



About This Document

Purpose

This document describes System Overview, Features, Package Information, Pin Information, Electrical Specification, AON Subsystem, CH Subsystem, SP Subsystem, AP Subsystem, Audio Subsystem and Camera Subsystem of UIS7885.

Intended Audience

This document is intended primarily for UIS7885 R&D and testing staff. Personnel must be familiar with the basic Knowledge of UNISOC Chip.

Acronyms and Abbreviations

Acronym and Abbreviation	Full Name
AC	Alternating Current
BGA	Ball Grid Array Package
CH	Context Hub
DC	Direct Current
ESD	Electro Static discharge
PGA	Pin Grid Array Package
SIM	Subscriber Identity Module
SP	System Processor

Symbol Conventions

The symbols that may be found in this guide are defined in the following table.

Symbol	Description
 NOTE	Calls attention to important information, best practices and tips. NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration.

Symbol	Description
	Calls attention to error-prone operations. CAUTION is used to address information not related to personal injury, equipment damage, and environment deterioration.
	Calls attention to irreversible operations. WARNING is used to address information not related to personal injury and environment deterioration.

Change History

Issue	Date	Description
V1.8	13-Jan-2026	Update 3.5 Pin Multiplexed Function List , 3.6 Pin Status and GPIOs
V1.7	17-Jul-2025	Update Figure 1-1
V1.6	16-Jul-2025	Update 4.1.3 ESD Characteristics
V1.5	12-Jun-2025	<ul style="list-style-type: none"> • Add 5.26 CH I3C • Update the voltage value of DCDC GEN0 • Update 1.4.3 Video Codec. • Update 10.1.2 DCAM Block. • Modify Camera's Specification.
V1.4	22-May-2024	Update 1.4.3 Multimedia Features Camera feature.
V1.3	07-Jul-2023	Update Table 2-1 and Figure 2-3 .
V1.2	29-Jun-2023	Update 3.9.4 SPSEBC2_24X_VL/HL .
V1.1	06-Mar-2023	<ul style="list-style-type: none"> • Modify Modem's Specification. • Modify Uart mode with software/hardware flow control. • Optimize description of this issue. • Modify Video Codec's Specification • RFCTL 0~15 Renamed RFCTL 0(G0) ~15 (G0) • RFCTL 16~31 Renamed RFCTL 0 (G1)~15 (G1)
V1.0	01-Aug-2022	This issue is the first official release.

Keywords

UIS7885, Datasheet, DeviceSpecification

Contents

1 System Overview	1
1.1 General Description.....	1
1.2 Functional Block Diagram.....	1
1.3 Highlighted Features.....	2
1.4 Features	3
1.4.1 Platform Features	3
1.4.2 Modem Features	4
1.4.3 Multimedia Features	6
2 Package Information	9
2.1 Basic Information	9
2.2 Top Marking Definition.....	9
2.3 Package Outline.....	10
2.4 Reflow Profile	12
2.5 Device Moisture Sensitivity Level	12
2.6 Thermal Characteristics	13
3 Pin Definition.....	14
3.1 BGA Pin out	14
3.2 Pin Symbol Description.....	14
3.3 Pin List	17
3.4 Pin Description	17
3.5 Pin Multiplexed Function List.....	39
3.6 Pin Status and GPIOs	53
3.7 Pin Register Descriptions	60
3.8 Strapping Pins.....	60
3.9 Pad Information	60
3.9.1 SPSCBC2_8X_VL/HL.....	61
3.9.2 SPSCBC2_8X_SR_VL/HL.....	62
3.9.3 SPSCBC2_8X_W_VL/HL	64
3.9.4 SPSEBC2_24X_VL/HL	66
3.9.5 PDWUWSWCDGSDE_V/H.....	68
3.9.6 SPSEBC2_SWP_HL/VL.....	69
4 Electrical Specification	77
4.1 DC Specification	77
4.1.1 Absolute Maximum Ratings	77
4.1.2 Recommended Operating Conditions.....	77
4.1.3 ESD Characteristics.....	79
4.1.4 DC Characteristics.....	79
4.2 AC Characteristics	81

4.3 Performance Specifications	83
4.3.1 LVDSRF	83
4.3.2 SOC Phase-Locked- Loop (PLL)	84
4.3.3 Thermal Sensor	85
4.3.4 USB 2.0 PHY	85
4.3.5 USB3.1/DP ComboPHY	86
4.3.6 MIPI CSI	87
4.3.7 MIPI C-PHY CSI	88
4.3.8 MIPI DSI	89
4.3.9 MIPI MPHY4.1	90
4.3.10 PCIEGEN3	92
4.3.11 Automatic Power Control DAC	94
4.3.12 CLK26M Buffer	95
4.3.13 RCO	95
4.3.14 DDC	95
5 AON Subsystem	97
5.1 General Description	97
5.1.1 Overview	97
5.1.2 Functional Block Diagram	97
5.2 AON Subsystem Address Map	97
5.2.1 Peripheral Address Map	97
5.2.2 IRAM Address Map	101
5.3 Interrupt List	102
5.3.1 INTC Interrupt Source List	102
5.3.2 EIC Interrupt List	110
5.4 ADI Channel List	112
5.5 Thermal Sensor List	113
5.6 MailBox Channel List	115
5.7 Analog-Digital Interface	115
5.7.1 Overview	115
5.7.2 Features	115
5.7.3 Function Description	116
5.7.4 Control Registers	126
5.7.5 Application Notes	126
5.8 APB_BUSMON	127
5.8.1 Overview	127
5.8.2 Features	128
5.8.3 Function Description	128
5.8.4 Control Registers	130
5.8.5 Application Notes	130
5.9 EFuse	132

5.9.1 Overview	132
5.9.2 Features	132
5.9.3 Control Registers	132
5.10 DJTAG	133
5.10.1 Overview	133
5.10.2 Features	133
5.10.3 Function Description	133
5.10.4 Control Registers	141
5.10.5 Application Notes	141
5.11 EIC	145
5.11.1 Overview	145
5.11.2 Features	146
5.11.3 Function Description	147
5.11.4 Control Registers	147
5.11.5 Application Notes	148
5.12 GPIO	150
5.12.1 Overview	150
5.12.2 Features	150
5.12.3 Function Description	150
5.12.4 Control Registers	151
5.12.5 Application Notes	151
5.13 IIS_MST	151
5.13.1 Overview	151
5.13.2 Features	151
5.13.3 Function Description	152
5.13.4 Control Registers	161
5.13.5 Application Notes	161
5.14 INTC	166
5.14.1 Overview	166
5.14.2 Features	166
5.14.3 Function Description	167
5.14.4 Control Registers	169
5.14.5 Application Notes	169
5.15 KPD Module	169
5.15.1 Overview	169
5.15.2 Features	170
5.15.3 Function Description	170
5.15.4 Control Registers	174
5.15.5 Application Notes	175
5.16 MailBox	175
5.16.1 Overview	175
5.16.2 Features	176

5.16.3 Function Description	176
5.16.4 Control Registers	180
5.16.5 User Lock	180
5.16.6 Sts_busy	180
5.16.7 Priority Configurable	180
5.16.8 IP Work Flow Description	181
5.16.9 Application Notes	182
5.17 PWM	183
5.17.1 Overview	183
5.17.2 Features	183
5.17.3 Function Description	183
5.17.4 Control Registers	186
5.17.5 Application Notes	186
5.18 SCC	187
5.18.1 Overview	187
5.18.2 Features	187
5.18.3 Function Description	188
5.18.4 Control Registers	190
5.18.5 Application Notes	190
5.19 Spinlock	190
5.19.1 Overview	190
5.19.2 Features	190
5.19.3 Function Description	191
5.19.4 Control Registers	193
5.20 SYS_FRT	193
5.20.1 Overview	193
5.20.2 Features	193
5.20.3 Function Description	193
5.20.4 Control Registers	194
5.20.5 Application Notes	194
5.21 SYST	194
5.21.1 Overview	194
5.21.2 Features	195
5.21.3 Function Description	195
5.21.4 Control Registers	195
5.21.5 Application Notes	195
5.22 THM_TOP	196
5.22.1 Overview	196
5.22.2 Features	196
5.22.3 Control Registers	196
5.22.4 Application Notes	196
5.23 Tmr_async_top	197

5.23.1 Overview	197
5.23.2 Features	197
5.23.3 Function Description	198
5.23.4 Control Registers	198
5.23.5 Application Notes	199
5.24 USB2.0	199
5.24.1 Features	199
5.24.2 Introduction	200
5.24.3 Functional Description	200
5.24.4 Control Registers	201
5.24.5 Operation in Peripheral mode	202
5.24.6 Operation in Host mode	205
5.25 Watchdog	207
5.25.1 Overview	207
5.25.2 Features	207
5.25.3 Function Description	207
5.25.4 Control Registers	208
5.25.5 Application Notes	208
5.26 CH I3C	209
5.26.1 Overview	209
5.26.2 Features	209
5.26.3 Function Description	209
5.26.4 Control Registers	217
5.26.5 Application Notes	217
6 CH Subsystem	222
6.1 Functional Block Diagram	222
6.2 Features	222
6.3 System Addr mapping	223
6.4 System Interrupt	223
6.5 System Low power Design	225
6.5.1 Power Scheme	225
6.5.2 Boot Information	225
6.6 DMA controller	226
6.6.1 Overview	226
6.6.2 Features	227
6.6.3 Function Descriptions	227
6.6.4 Control Registers	228
6.6.5 Application Notes	228
6.7 Sensor Time Stamp	229
6.7.1 Overview	229
6.7.2 Function Descriptions	229

6.7.3 Application Notes	229
7 SP Subsystem	230
7.1 Functional Block Diagram.....	230
7.2 Features	230
7.3 System Addr mapping	231
7.4 System Interrupt	231
7.5 System Low power Design.....	233
7.5.1 Power Scheme	234
7.5.2 Boot Information	234
8 AP Subsystem.....	235
8.1 Overview	235
8.2 Features	235
8.3 AP Memory Map	236
8.4 Interrupt.....	237
8.5 AP Clock Control Register	238
8.6 AP AHB Control Register.....	239
8.7 AP APB Control Register	239
8.8 DMA.....	239
8.8.1 Overview	239
8.8.2 Features	239
8.8.3 Function Description	240
8.8.4 Control Registers.....	246
8.8.5 Application Notes	246
8.9 SDIO0/SDIO1/SDIO2	246
8.10 eMMC Host.....	247
8.10.1 Overview	247
8.10.2 Features	247
8.10.3 Function Description	248
8.10.4 Control Registers.....	255
8.10.5 Application Notes	256
8.11 UART Module	258
8.11.1 Overview	258
8.11.2 Features.....	258
8.11.3 Function Description	259
8.11.4 Control Registers.....	267
8.11.5 Application Notes	267
8.12 SPI_HS	271
8.12.1 Overview	271
8.12.2 Features	271
8.12.3 Function Description	271
8.12.4 Control Registers.....	289

8.12.5 Application Notes	289
8.13 IIS	300
8.13.1 Overview	300
8.13.2 Features	300
8.13.3 Function Description	301
8.13.4 Control Registers	308
8.13.5 Application Notes	308
8.14 I2C	314
8.14.1 Overview	314
8.14.2 Features	315
8.14.3 Function Description	315
8.14.4 Control Registers	324
8.14.5 Application Notes	324
8.15 UFS	328
8.15.1 Overview	328
8.15.2 Features	328
9 Audio Subsystem	330
9.1 General Description	330
9.1.1 Overview	330
9.1.2 Audio Path Features	330
9.1.3 System Addr mapping	331
9.1.4 System Interrupt	332
9.1.5 System Power Design	335
9.2 Processor	337
9.2.1 Overview	337
9.2.2 Features	337
9.3 Voice Trigger	337
9.3.1 Overview	337
9.3.2 Features	337
9.3.3 Function Description	337
9.4 Audio codec digital parts	338
9.4.1 Overview	338
9.4.2 Features	338
9.4.3 Function Description	339
9.4.4 Control Registers	341
9.4.5 Application Notes	341
9.5 VBC interface	341
9.5.1 Overview	341
9.5.2 Features	342
9.5.3 Function Description	343
9.5.4 Control Registers	356

9.5.5 Application Notes	356
9.6 SRC_48K.....	382
9.6.1 Overview	382
9.6.2 Features	382
9.6.3 Function Description	382
9.6.4 Control Registers.....	383
9.6.5 Application Notes	383
9.7 MCDT	384
9.7.1 Overview	384
9.7.2 Features	384
9.7.3 Function Description	384
9.7.4 Control Registers.....	388
9.7.5 Application Notes	388
9.8 PDM	389
9.8.1 Overview	389
9.8.2 Features	389
9.8.3 Function Description	389
9.8.4 Control Registers.....	393
9.9 DMA.....	393
9.9.1 Overview	393
9.9.2 Features	393
9.9.3 Function Description	394
9.9.4 Control Registers.....	403
9.9.5 Application Notes	403
9.10 IIS	403
9.10.1 Overview	403
9.10.2 Features	404
9.10.3 Function Description	404
9.10.4 Control Registers.....	412
9.10.5 Application Notes	412
9.11 UART.....	418
9.11.1 Overview	418
9.11.2 Features.....	419
9.11.3 Function Description	419
9.11.4 Control Registers.....	422
9.11.5 Application Notes	423
9.12 Spinlock.....	424
9.12.1 Overview	424
9.12.2 Features	424
9.12.3 Function Description	424
9.12.4 Control Registers.....	426
9.13 Watchdog timer.....	427

9.13.1 Overview	427
9.13.2 Features	427
9.13.3 Function Description	427
9.13.4 Control Registers	427
9.13.5 Application Notes	428
9.14 VAD	428
9.14.1 Overview	428
9.14.2 Features	429
9.14.3 Function Description	429
9.14.4 Control Registers	429
9.15 TDM	430
9.15.1 Overview	430
9.15.2 Features	430
9.15.3 Function Description	431
9.15.4 Control Registers	435
9.15.5 Application Notes	435
10 MultiMedia Subsystem	436
10.1 Camera	436
10.1.1 General Description	436
10.1.2 DCAM Block	438
10.1.3 ISP Block	441
10.1.4 VDSP Block	444
10.1.5 JPG	451
10.1.6 Interface	453
11 AI Subsystem	480
11.1 System Overview	480
11.1.1 Functional Block Diagram	480
11.1.2 Features	480
11.1.3 System Low power Design	481
11.2 Neural network accelerator	482
11.2.1 Key Features	482
11.2.2 Supported Layers	483
11.2.3 Hardware Processing Order	484
12 iSE System Design Specification	486
12.1 Overview	486
12.2 Features	486
13 Reference Documents	487

List of Figures

Figure 1-1 Functional Block Diagram of UIS7885	2
Figure 2-1 Silk screen printing.....	9
Figure 2-2 Package outline	10
Figure 2-3 Package parameters	11
Figure 2-4 Reflow profile	12
Figure 3-1 BGA Pin out	14
Figure 3-2 Diagram of PDWUWSWCDGSDE_V/H Pad	68
Figure 3-3 SWP data transmission.....	72
Figure 3-4 SWPIO Voltage definition for the signal S1	73
Figure 3-5 SWPIO Current definition for the signal S2	73
Figure 3-6 SWPIO diagram	74
Figure 3-7 SWPIO current generation	75
Figure 4-1 Test circuit of an I/O pin.....	82
Figure 5-1 ADI Block Diagram	117
Figure 5-2 FSM State transmission diagram.....	122
Figure 5-3 Write Operation.....	123
Figure 5-4 Read Operation.....	124
Figure 5-5 ADI Slave	125
Figure 5-6 DJATG architecture.....	133
Figure 5-7 FSM.....	134
Figure 5-8 bypass circuit.....	137
Figure 5-9 idcode/usercode circuit.....	137
Figure 5-10 instruction register.....	138
Figure 5-11 instruction circuit.....	138
Figure 5-12 scan chain circuit.....	139
Figure 5-13 scan chain behavior	140
Figure 5-14 bypass mode control flow	142
Figure 5-15 idcode/usercode mode control flow	143

Figure 5-16 chain scan read mode control flow	144
Figure 5-17 chain scan write mode control flow	145
Figure 5-18 Software control flow	148
Figure 5-19 Software control flow(system wakeup).....	149
Figure 5-20 IIS module diagram	152
Figure 5-21 IIS-compatible format	153
Figure 5-22 IIS-compatible format with dummy bit.....	153
Figure 5-23 MSB-justified format	153
Figure 5-24 MSB-justified format with dummy bit.....	154
Figure 5-25 Synchronized format	154
Figure 5-26 IIS works in tx_only mode case1	155
Figure 5-27 IIS works in tx_only mode case2	155
Figure 5-28 IIS works in rx_only mode.....	156
Figure 5-29 {NG_RX, NG_RX} is 2'b00 or 2'b11.....	157
Figure 5-30 {NG_RX, NG_RX} is 2'b01 or 2'b10.....	158
Figure 5-31 IIS clock path	159
Figure 5-32 Sampling data at negedge of IIS_SCK.....	160
Figure 5-33 Sampling data at posedge of IIS_SCK.....	160
Figure 5-34 implementation of sampling data at negedge of IIS_SCK.....	160
Figure 5-35 implementation of sampling data at posedge of IIS_SCK	161
Figure 5-36 implementation of sampling data at posedge of IIS_SCK	161
Figure 5-37 iis_mst (tx_fifo filled with 3 data).....	166
Figure 5-38 iis (tx_fifo filled with 3 data)	166
Figure 5-39 Function diagram.....	168
Figure 5-40 Interrupt source	169
Figure 5-41 Keypad function diagram	170
Figure 5-42 Key status logic state machine	171
Figure 5-43 Single-key mode.....	171
Figure 5-44 Long-key mode	172
Figure 5-45 Multi-key mode.....	172
Figure 5-46 GPIO connection with old KPD (r3p0).....	173

Figure 5-47 r4p0 KPD modification strategy	173
Figure 5-48 Keypad detected one “error” status	174
Figure 5-49 software control flow	175
Figure 5-50 Mail Box top diagram	176
Figure 5-51 Mail Box data flow	177
Figure 5-52 Mail description.....	177
Figure 5-53 Out Box diagram	179
Figure 5-54 Priority adjustment diagram	181
Figure 5-55 PWM Block Diagram.....	184
Figure 5-56 PWM Timing Diagram.....	185
Figure 5-57 PWM delay not support.....	185
Figure 5-58 PWM delay support.....	186
Figure 5-59 SCC Logic Block Diagram	188
Figure 5-60 SCC_TYPE Architecture.....	189
Figure 5-61 Write waveform for Handshake	190
Figure 5-62 Read waveform for Handshake	190
Figure 5-63 SpinLock Block Diagram.....	191
Figure 5-64 FSM Diagram.....	191
Figure 5-65 Program Flow Diagram.....	192
Figure 5-66 Interface Architecture Diagram	193
Figure 5-67 Application Diagram	194
Figure 5-68 timer architecture.....	198
Figure 5-69 I3C Data Transfer – ACK by Slave.....	212
Figure 5-70 I3C Data Transfer – NACK by Slave.....	212
Figure 5-71 T-Bit When Slave Ends Read and Master Generates STOP	213
Figure 5-72 T-Bit When Slave Ends Read and Master Generates Repeated START	213
Figure 5-73 T-Bit When Slave and Master Agree to Continue Read Message.....	214
Figure 5-74 T-Bit When Master Ends Read with Repeated START and STOP	214
Figure 5-75 T-Bit When Master Ends Read via Repeated START and Further Transfer	215
Figure 6-1 The Functional Block Diagram of CH(Context Hub).....	222
Figure 6-2 AHB DMA Arch.....	226

Figure 6-3 SENSOR TS Arch	229
Figure 7-1 The Functional Block Diagram of SP(System Processor)	230
Figure 8-1 Architecture of AP SYS.....	235
Figure 8-2 eMMC System Overview	248
Figure 8-3 eMMC host System Block Diagram	249
Figure 8-4 eMMC PHY CLK/CMD/DAT Timing Relation	250
Figure 8-5 eMMC PHY CLK Logic	250
Figure 8-6 eMMC PHY CLK, CMD/DAT Write Logic	251
Figure 8-7 eMMC PHY CMD/DAT Read Logic	252
Figure 8-8 ADMA3 example	256
Figure 8-9 tuning sequence	257
Figure 8-10 the fine tuning point	258
Figure 8-11 Diagram of UART	259
Figure 8-12 Software Flow Control Processing Flow	262
Figure 8-13 Software Flow Control Data Stream (Mode0/Model1).....	262
Figure 8-14 Diagram of tx flow control.....	263
Figure 8-15 Diagram of rx flow control.....	263
Figure 8-16 Uart_tx start bit pulse and uart_tx receive control flow	264
Figure 8-17 Uart_rx flow control.....	265
Figure 8-18 UART timing parameter	266
Figure 8-19 IrDA transmit timing	266
Figure 8-20 Uart automatic baudrate detection flow	267
Figure 8-21 Uart mode with hardware flow control	268
Figure 8-22 Uart with DMA mode with hardware flow control	269
Figure 8-23 Uart mode with software flow control	270
Figure 8-24 Uart with DMA mode with software flow control	270
Figure 8-25 Diagram of SPI Master.....	272
Figure 8-26 Diagram of SPI Slave.....	272
Figure 8-27 MICROWIRE/PLUS Timing (CPOL=0, CPHA=0).....	273
Figure 8-28 Alternated Phase SK Clock Timing (CPOL=0, CPHA=1).....	273
Figure 8-29 MICROWIRE/PLUS Timing (CPOL=1, CPHA=0).....	273

Figure 8-30 MICROWIRE/PLUS Timing (CPOL=1, CPHA=1)	274
Figure 8-31 Alternated Phase SK Clock Timing (CPOL=0, CPHA=1).....	275
Figure 8-32 4-wire Serial Interface (s8) Timing	275
Figure 8-33 3-wire Serial Interface Timing 1	276
Figure 8-34 3-wire Serial Interface Timing 2	277
Figure 8-35 spi_cd_mode.....	277
Figure 8-36 spi_cd_mode.....	278
Figure 8-37 spi_3w_mode	278
Figure 8-38 2-data-line interfaces of writing data Timing.....	279
Figure 8-39 2-data-line interfaces of reading data Timing	279
Figure 8-40 2-data-line interfaces of writing pixel data Timing	280
Figure 8-41 RGB565 1 pixel/transition	280
Figure 8-42 RGB666 1 pixel/transition	281
Figure 8-43 RGB888 1 pixel/transition	281
Figure 8-44 RGB565 1 pixel/transition(set DATA_LINE2_SW to 1).....	282
Figure 8-45 TX Dummy waveform	284
Figure 8-46 RX Dummy waveform.....	284
Figure 8-47 SPI Slave Clock.....	285
Figure 8-48 SPI Slave TX and RX Clock	285
Figure 8-49 SPI Slave Reset	286
Figure 8-50 TX logic structure.....	286
Figure 8-51 TX timing chart	287
Figure 8-52 RX Logic structure	287
Figure 8-53 RX timing chart.....	288
Figure 8-54 SPI_HS Slave Clock Path	288
Figure 8-55 Slave Clock Delay Path.....	289
Figure 8-56 8 bit read process timing	292
Figure 8-57 24/32 bit read process timing	293
Figure 8-58 2 transactions of read transfer process	293
Figure 8-59 two data line mode timing.....	294
Figure 8-60 IIS module diagram.....	301

Figure 8-61 IIS-compatible format	302
Figure 8-62 IIS-compatible format with dummy bit.....	302
Figure 8-63 MSB-justified format	302
Figure 8-64 MSB-justified format with dummy bit.....	302
Figure 8-65 Synchronized format	303
Figure 8-66 Long frame sync (Shown with 8-bit sample)	304
Figure 8-67 Short frame sync (Shown with 16-bit sample).....	304
Figure 8-68 Multi-slot operation with two slots and 8-bit compounded samples	305
Figure 8-69 Sampling data at negedge of IIS_SCK.....	307
Figure 8-70 Sampling data at posedge of IIS_SCK.....	307
Figure 8-71 implementation of sampling data at negedge of IIS_SCK.....	307
Figure 8-72 implementation of sampling data at posedge of IIS_SCK	308
Figure 8-73 implementation of sampling data at posedge of IIS_SCK	308
Figure 8-74 I2C system diagram.....	315
Figure 8-75 Data transfer on the I2C-bus	316
Figure 8-76 Clock synchronization during the arbitration procedure.....	317
Figure 8-77 I2C timing diagram	318
Figure 8-78 DMA transmission mode.....	319
Figure 8-79 Architecture	320
Figure 8-80 software control flow	325
Figure 9-1 Power Arch.....	336
Figure 9-2 CEVA-TL420 MEM.....	337
Figure 9-3 DMIC and AMIC Data Flow.....	338
Figure 9-4 Audio clock structure	340
Figure 9-5 IIS interface timing.....	341
Figure 9-6 DAC FIFO.....	343
Figure 9-7 DAC FIFO Data format 1.....	344
Figure 9-8 DAC FIFO Data format 2.....	344
Figure 9-9 DAC FIFO Data format 3.....	344
Figure 9-10 DAC FIFO Data format 4.....	344
Figure 9-11 ADC FIFO	346

Figure 9-12 Audio Play FIFO	347
Figure 9-13 Audio Record FIFO	348
Figure 9-14 IIS timing 1	348
Figure 9-15 IIS timing 2	349
Figure 9-16 PCM timing	349
Figure 9-17 TDM Connection	350
Figure 9-18 TDM 8/4/2 Channel timing	351
Figure 9-19 TDM 4 Channel timing 1	352
Figure 9-20 TDM 4 Channel timing 2	352
Figure 9-21 TDM 4 Channel timing 3	353
Figure 9-22 TDM 4 Channel timing 4	353
Figure 9-23 TDM 4 Channel timing 5	354
Figure 9-24 TDM 4 Channel timing 6	354
Figure 9-25 TDM 4 Channel timing 7	355
Figure 9-26 TDM 4 Channel timing 8	355
Figure 9-27 TDM 4 Channel timing 9	356
Figure 9-28 MCDT connect relation	384
Figure 9-29 FIFO control flow 1	386
Figure 9-30 DMIC interface timing diagram	389
Figure 9-31 PDM IIS0 DATA	392
Figure 9-32 PDM IIS1 DATA	392
Figure 9-33 PDM IIS2 DATA	392
Figure 9-34 PDM IIS DATA	392
Figure 9-35 pdm data 24bit slot length 32bit	393
Figure 9-36 DMA data path	393
Figure 9-37 DMA data structure for standard channel	395
Figure 9-38 DMA data structure for full channel	395
Figure 9-39 DMA arbitration state machine	396
Figure 9-40 DMA data movement example-1	397
Figure 9-41 DMA data movement example-2	397
Figure 9-42 DMA transfer examples	398

Figure 9-43 DMA channel execution flow	400
Figure 9-44 DMA link list.....	401
Figure 9-45 DMA one channel start other channel	402
Figure 9-46 channel Y is idle while channel X start request	402
Figure 9-47 channel Y is busy while channel X start request	403
Figure 9-48 IIS module diagram.....	405
Figure 9-49 IIS-compatible format	405
Figure 9-50 IIS-compatible format with dummy bit.....	406
Figure 9-51 MSB-justified format	406
Figure 9-52 MSB-justified format with dummy bit.....	406
Figure 9-53 Synchronized format	407
Figure 9-54 Long frame sync (Shown with 8-bit sample).....	408
Figure 9-55 Short frame sync (Shown with 16-bit sample).....	408
Figure 9-56 Multi-slot operation with two slots and 8-bit compounded samples	409
Figure 9-57 Sampling data at negedge of IIS_SCK.....	411
Figure 9-58 Sampling data at posedge of IIS_SCK.....	411
Figure 9-59 implementation of sampling data at negedge of IIS_SCK.....	411
Figure 9-60 implementation of sampling data at posedge of IIS_SCK	412
Figure 9-61 implementation of sampling data at posedge of IIS_SCK	412
Figure 9-62 Software Flow Control Data Stream (Mode0/Mode1).....	421
Figure 9-63 UART timing parameter.....	422
Figure 9-64 IrDA transmit timing	422
Figure 9-65 Spinlock.....	424
Figure 9-66 take and release spinlock.....	425
Figure 9-67 spinlock take and release flow	426
Figure 9-68 TDM Top Diagram	431
Figure 9-69 TDM basic timing	432
Figure 9-70 TDM4 data16bit slot length32bit	432
Figure 9-71 TDM4 data16bit slot length16bit	433
Figure 9-72 TDM4 data24bit slot length32bit	433
Figure 9-73 TDM4 data16bit slot length32bit	434

Figure 10-1 Camera System Block Diagram	437
Figure 10-2 DCAM Block Diagram	440
Figure 10-3 ISP Block Diagram.....	441
Figure 10-4 VDSP Block Diagram	444
Figure 10-5 VDSP Core Block Diagram.....	446
Figure 10-6 VDMA module Diagram.....	447
Figure 10-7 VAU module Diagram.....	448
Figure 10-8 UART module Diagram	450
Figure 10-9 MAILBOX module Diagram	451
Figure 10-10 JPG module Diagram	452
Figure 10-11 C-PHY High-Speed Data transmission in Burst	454
Figure 10-12 D-PHY High-Speed Data transmission in Burst	454
Figure 11-1 AI system architecture	480
Figure 11-2 AI sys power architecture	481
Figure 11-3 Processing Order of Layers in Hardware	485

List of Tables

Table 2-1 Package Information	9
Table 2-2 MSL ratings summary	12
Table 2-3 Thermal characteristics.....	13
Table 3-1 Definition of pin symbols.....	14
Table 3-2 Pin description.....	17
Table 3-3 Pin Multiplexed Function List.....	39
Table 3-4 Truth Table of SPSCBC2_8X_VL/HL Pad.....	61
Table 3-5 Driver Strength Select Function of SPSCBC2_8X_VL/HL Pad	61
Table 3-6 DC Parameter of SPSCBC2_8X_VL/HL Pad.....	62
Table 3-7 AC Parameter of SPSCBC2_8X_VL/HL Pad.....	62
Table 3-8 Truth Table of SPSCBC2_8X_SR_VL/HL Pad	62
Table 3-9 Driver Strength Select Function of SPSCBC2_8X_SR_VL/HL Pad	63
Table 3-10 DC Parameter of SPSCBC2_8X_SR_VL/HL Pad.....	63
Table 3-11 AC Parameter of SPSCBC2_8X_SR_VL/HL Pad	64
Table 3-12 Truth Table of SPSCBC2_8X_W_VL/HL Pad	64
Table 3-13 Driver Strength Select Function of SPSCBC2_8X_W_VL/HL Pad.....	65
Table 3-14 DC Parameter of SPSCBC2_8X_W_VL/HL Pad.....	65
Table 3-15 AC Parameter of SPSCBC2_8X_W_VL/HL Pad	65
Table 3-16 Truth Table of SPSEBC2_24X_VL/HL Pad	66
Table 3-17 Driver Strength Select Function of SPSEBC2_24X_VL/HL Pad.....	66
Table 3-18 DC Parameter of SPSEBC2_24X_VL/HL Pad.....	67
Table 3-19 AC Parameter of SPSEBC2_24X_VL/HL Pad	67
Table 3-20 Truth Table of PDWUWSWCDGSDE_V/H Pad.....	68
Table 3-21 Driver Strength Select Function of PDWUWSWCDGSDE_V/H Pad	69
Table 3-22 SWPIO pin list1	70
Table 3-23 SWPIO pin list2	70
Table 3-24 SWPIO cell list.....	71
Table 3-25 SWPIO electrical Specifications	72

Table 3-26 SWPIO delay path	74
Table 3-27 SWPIO application setting	75
Table 3-28 Driver Strength Select Function of SPSEBC2_SWP_HL/VL Pad	76
Table 4-1 Absolute maximum ratings	77
Table 4-2 Recommended operating conditions	77
Table 4-3 ESD characteristics	79
Table 4-4 DC Characteristics	79
Table 4-5 Standard input, output and I/O pin AC characteristics	82
Table 4-6 System Clock AC characteristics	82
Table 4-7 LVDS TX DC specifications	83
Table 4-8 LVDS RX DC specifications	83
Table 4-9 LVDS TX AC specifications	83
Table 4-10 LVDS RX AC specifications	84
Table 4-11 Performance of the phase-locked loop	84
Table 4-12 Performance of the Thermal sensor	85
Table 4-13 Main performance of the USB	85
Table 4-14 Main performance of the USB3.1/DP	86
Table 4-15 Main performance of the MIPI CSI	87
Table 4-16 Main performance of the MIPI DSI	89
Table 4-17 Main performance of the MIPI MPHY	90
Table 4-18 Main performance of the PCIEGEN3 2.5GT/s	92
Table 4-19 Main performance of the PCIEGEN3 5GT/s	92
Table 4-20 Main performance of the PCIEGEN3 8GT/s	93
Table 4-21 Performance of the APC DAC	94
Table 4-22 Performance of the 26MHz Buffer	95
Table 4-23 Performance of the RCO	95
Table 4-24 Main performance of the DDC	95
Table 5-1 bus_match_int_req_raw interrupt generation	130
Table 5-2 Debug bus list	131
Table 5-3 instruction table	138
Table 5-4 Function table	140

Table 5-5 IIS data format memory map	154
Table 5-6 IIS configure for fraction clock divide	163
Table 5-7 iis_clk configure suggestion.....	164
Table 5-8 SCC signals description	188
Table 6-1 The ch Address map	223
Table 6-2 ch_sys interrupt	224
Table 6-3 CH STAR MCU power states information.....	225
Table 6-4 AON DMA Request List	228
Table 7-1 The sp Address map	231
Table 7-2 sp_sys interrupt	231
Table 7-3 sp_STAR MCU power states information.....	234
Table 8-1 AP memory map	236
Table 8-2 AP interruption list	237
Table 8-3 Signals List.....	252
Table 8-4 Signals List.....	253
Table 8-5 1 pixel mode.....	281
Table 8-6 2/3 pixel mode.....	282
Table 8-7 RGB888 mode APB wdata format	283
Table 8-8 RGB888 mode TXFIFO wdata format.....	283
Table 8-9 RGB666 mode APB wdata format	283
Table 8-10 RGB666 mode TXFIFO wdata format.....	283
Table 8-11 RGB565 mode APB wdata format	284
Table 8-12 RGB565 mode TXFIFO wdata format.....	284
Table 8-13 IIS data format memory map	303
Table 8-14 PCM data format for master mode	305
Table 8-15 PCM data format for slave mode	305
Table 8-16 PCM multiple slot and PCM cycle configuration suggestion	306
Table 8-17 Exceptions for PCM multiple slot configuration	306
Table 8-18 IIS configure for fraction clock divide	311
Table 8-19 iis_clk configure suggestion.....	312
Table 8-20 I2C interrupt.....	315

Table 8-21 I2C timing diagram	318
Table 8-22 Channel name and number	322
Table 9-1 Audio sys address	331
Table 9-2 Interrupts of the audio sys	333
Table 9-3 DVFS Range	336
Table 9-4 Power states in AUDCP SYS	336
Table 9-5 DAC SRC mode	345
Table 9-6 ADC SRC mode	346
Table 9-7 IIS master mode configure	349
Table 9-8 TDM master mode configure	351
Table 9-9 Adc paths IIS mux	362
Table 9-10 Dac paths IIS clk and lrck select	364
Table 9-11 IIS do select	365
Table 9-12 cycles for one input data.....	382
Table 9-13 cycles for one output data.....	383
Table 9-14 IIS data format memory map	407
Table 9-15 PCM data format for master mode	409
Table 9-16 PCM data format for slave mode	409
Table 9-17 PCM multiple slot and PCM cycle configuration suggestion	410
Table 9-18 Exceptions for PCM multiple slot configuration	410
Table 9-19 IIS configure for fraction clock divide	415
Table 9-20 iis_clk configure suggestion.....	416
Table 10-1 Cam Application Case	437
Table 10-2 MIPI PHY Description	440
Table 10-3 Combo PHY PAD list.....	453
Table 10-4 2p2lane D-PHY PAD list.....	455
Table 11-1 Power domain in AI SYS	482
Table 11-2 Power states in AI SYS.....	482
Table 11-3 Supported Layers.....	483

1 System Overview

1.1 General Description

UIS7885 is a high performance application processor with integrated 2G/3G/4G/5G modem.

It consists of one-core Arm® Cortex®-A76 up to 2.7GHz, three-core Arm® Cortex®-A76 up to 2.3GHz, 4-core Arm® Cortex®-A55 up to 2.1GHz. In addition, an extensive set of interfaces is included to interface to cameras, touch panel display and UFS/eMMC/SD cards.

The application processor equipped with Arm® Neon™ engine support the latest Open OS along with its demanding applications such as web browsing, email and games.

The multi-standard multi-media accelerators and advanced audio subsystem are also integrated to provide advance multimedia applications and service such as streaming audio and video.

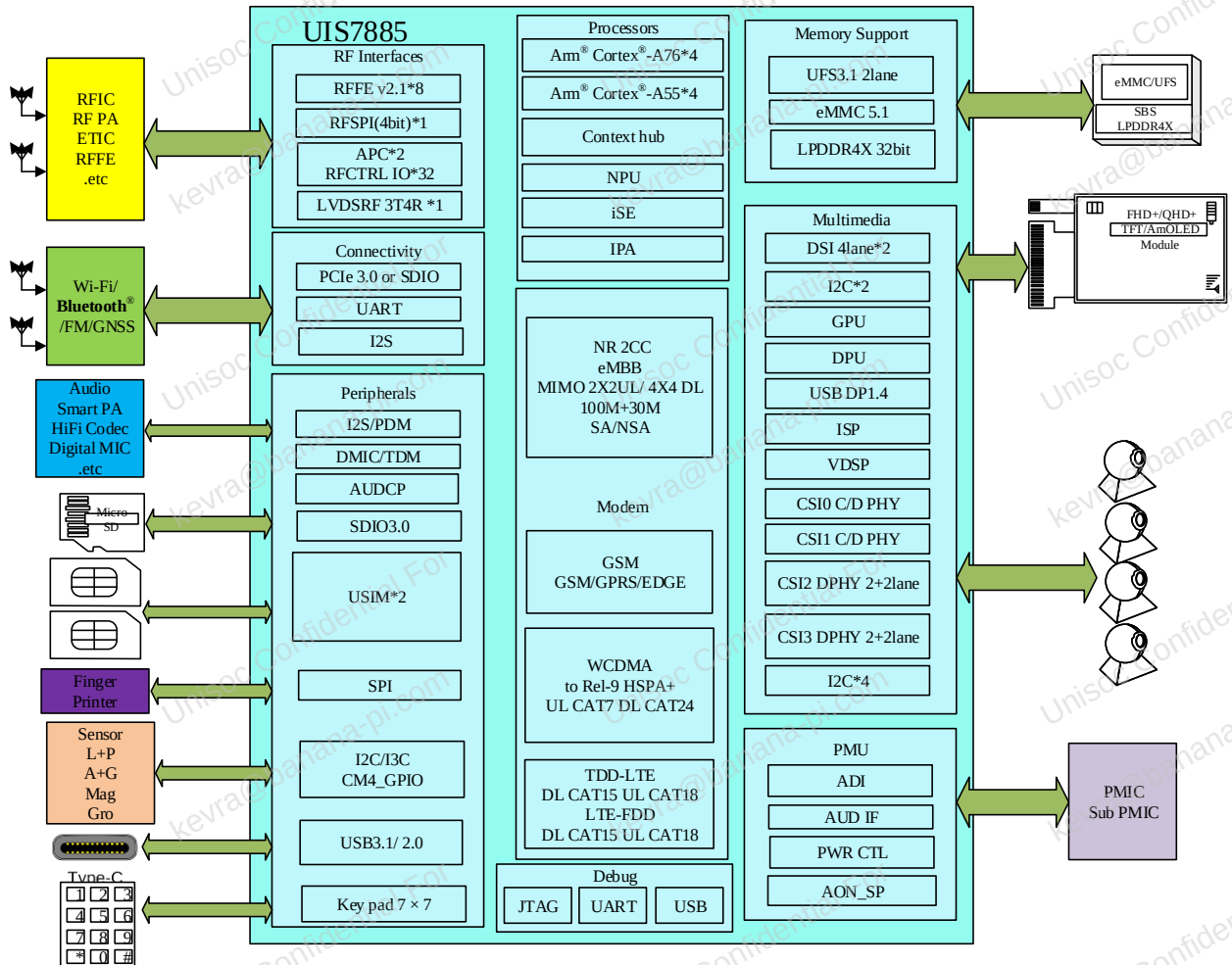
The high performance CPU and hardware accelerator are combined to provide modem subsystem capable of supporting NR FR1, LTE UL Cat18 and DL Cat15, Cat24 HSDPA and Cat7 HSUPA data rate, Class12 GPRS/EDGE.

The specially optimized architecture of UIS7885 can achieve high performance and low power for a lot of applications. Proprietary architectures and algorithms were developed for low power ASIC design and power management. Unique techniques are used for noise suppression/cancellation, echo suppression/cancellation algorithm. Overall, UIS7885 chipset presents a high performance-efficiency platform for 5G devices.

1.2 Functional Block Diagram

The following figure shows the chip-level functional block diagram of UIS7885.

Figure 1-1 Functional Block Diagram of UIS7885



1.3 Highlighted Features

- 6nm EUV process for best-in-class power-efficient 5G SOC chip
- One-core Arm® Cortex®-A76 up to 2.7GHz
- Three-core Arm® Cortex®-A76 up to 2.3GHz
- Four-core Arm® Cortex®-A55 up to 2.1GHz
- Shared 3MB L3 cache
- Arm® Mali™-G57, up to 850MHz
- IMG AX3596 up to 988MHz
- Support LPDDR4X, clock up to 2133MHz
- Support UFS3.1
- Support inter-band/Intra-band 2CC 100M + 30M
- FDD: DL Cat15 + UL Cat18(LTE only)
- TDD: DL Cat15 + UL Cat18(LTE only)

- EUTRA: DL Cat15, UL Cat18(EN-DC)
- Support QHD+@60fps/FHD+@120fps
- VP9: encoder to 1080p@30fps, decoder to profile0/profile2, 4k@30fps
- Single Camera up to 108M(9in1)
- Support Low Power Voice trigger

1.4 Features

1.4.1 Platform Features

AP subsystem

- One-core Arm® Cortex®-A76 up to 2.7GHz and three-core Arm® Cortex®-A76 up to 2.3GHz, each core with 64KB L1 I-cache, 64KB L1 D-cache and 256KB L2 cache
- Four-core Arm® Cortex®-A55 up to 2.1GHz, each core with 32KB L1 I-cache, 32KB L1 D-cache and 128KB L2 cache
- Shared 3MB L3 cache
- Arm® Neon™ multimedia processing engine with SIMDv2/VFPv4 ISA
- DVFS technology with operating voltage from 0.65V~0.95V

NPU Subsystem

- IMG AX3596 up to 988MHz
- Support a range of neural network operation such convolutions, activations, element-wise operations, pooling etc.
- Lossless weight data compression
- DRM security and scheduling for multiple hosts

VDSP Subsystem

VQ7 up to 1014MHz, 256KB Data RAM

Security Subsystem

Isolated execution environment

External memory interface

- Support eMMC 5.1 HS400
- Support LPDDR4X
- Support UFS3.1
- Supporting the following memory combinations:
 - eMCP(eMMC/LPDDR4X)
 - eMMC + LPDDR4X
 - uMCP(UFS3.1/LPDDR4X)
 - UFS3.1 + LPDDR4X
- eMMC flash is supported by eMMC controller, with the following features:

- Support 1.8V I/O
- Support 1.8 or 3.0V VCC
- Support 1/4/8-bit device
- Compliance to eMMC5.1
- LPDDR4X is supported by DDR Controller, with the features list below:
 - Support 2×16 -bit LPDDR4X
 - Support dual channel
 - Support up to 32GB density
 - Support up to 2 CS
 - Support up to 32-bits data width
 - Clock up to 2133MHz
- UFS3.1 MPHY Support the following feature:
 - Support MPHY Type 1 M-ports
 - Internal(Through SOC) and external(Through PAD) reference clock supported: 26MHz
 - 2 TX lane and 2 RX lane operation
 - High Speed G1/2/3/4, Mode A/B operation
 - Support 1.8V and 1.2V Analog Supply
 - Compliance to MPHY V4.1
 - Compliance to UniPro V1.8

Peripheral and Connectivity Interfaces

- Support 2-SIM cards, both 1.8V and 3.0V devices
- Support 3 SDIO 3.0 with 4bit data width
- Support USB 2.0/3.1 Gen2/ USB 3.0 OTG
- Support 8 UARTs, 2 UARTs support hardware flow control
- Support 4 SPIs, support both master and slave
- Support 6 I2S(PCM) for audio codec connection
- Support 3×3 keypad default up to 7×7
- Support 10 I2Cs up to 3.4Mbps, include 2 I3Cs
- Support more than 198 GPIO pins
- Support 3 PWM outputs
- Support SIM/USIM
- Support 6 digital MICs
- Support 1 PCIE 3.0 port with 1 lane

1.4.2 Modem Features

NR Baseband

- 3GPP Release 15
- Support NSA/ SA
- Support NSA NR 1CC + LTE 1CC/ NR 1CC + LTE 2CC
- Support inter-band/Intra-band 2CC 100M + 30M

- Support 4×4 MIMO
- Support 256 QAM for DL and UL
- Support Cryptographic and integrity algorithms: NIA0/ NIA1/ NIA2/ NIA3 and NEA0/ NEA1/ NEA2/ NEA3
- Support PWS on NR
- Support Network slice
- Support VoNR and EPS fallback

TD-LTE/LTE FDD Baseband

- 3GPP Release 15
- FDD: DL Cat15 + UL Cat18 (LTE only)
- TDD: DL Cat15 + UL Cat18 (LTE only)
- Support DL $2/4$ layers MIMO($2 \times 2 / 4 \times 2 / 8 \times 2 / 4 \times 4 / 8 \times 4$)
- Support 256QAM for UL and DL
- Support intra/inter-band 2CA/3CA DL
- Support intra-band continuous 2CA UL
- Support FDD + TDD CA
- Support VoLTE/ViLTE
- Support MFBI
- Support Short DRX/ Long DRX/ C-DRX
- Support TTI Bundling/SPS
- Cryptographic and integrity algorithms: EEA1/EIA1:SNOW-3G, EEA2/EIA2:AES, EEA3/EIA3:ZUC
- Support RoHC
- Support PWS(Public Warning System, including ETWS and CMAS) receiving
- Support EN-DC

WCDMA/HSDPA/HSUPA Baseband

- 3GPP release 9
- Support downlink Category 24 (64QAM, dual-cell HSDPA), throughput up to 42Mbps
- Support uplink Category 7 (16QAM, HSUPA), throughput up to 11Mbps
- Support CPC/enhanced cell-FACH/enhanced F-DPCH/eDRX
- Support SRBoHS, CELL & URA-PCH
- Support DL diversity and type3i receiver
- Support Fast Dormancy
- Cryptographic and integrity algorithms: UEA1/UIA1: KASUMI, UEA2/UIA2: SNOW-3G
- Support PWS(Public Warning System, including ETWS and CMAS)receiving

GSM/GPRS/EDGE Baseband

- GSM/GPRS/EDGE Release 6, Release 9 particularly for LTE inter-operation
- Support GSM850, GSM900, DCS1800, and PCS1900
- Support GPRS/EDGE class12
- Support GPRS service: class B

- Cryptographic algorithms: A5/1 A5/3, GEA1 GEA2 GEA3
- Support DL diversity receiver
- Support VAMOS I
- Support DARPI
- Support PWS(Public Warning System, including ETWS and CMAS) receiving

Multi-mode

- Support DSDS
- Support eSRVCC/aSRVCC/bSRVCC/mid-SRVCC
- Support PSHO/ CSFB/ Redirection

Voice Codec

- Support quad vocoders for AMR-NB, EFR, HR and FR
- Support AMR-WB/AMR-NB
- Support EVS-NB/WB/SWB*
- Support noise reduction
- Support echo suppression/echo cancellation

1.4.3 Multimedia Features

3D Graphics

- Arm® Mali™-G57 4cores, up to 850MHz
- Support OpenGL ES 3.2/3.1/3.0/2.0/1.1
- OpenCL 2.x
- Vulkan 1.1
- Support DVFS technology

Image(JPEG)

- Compliance with JPEGISO/IEC 10918-1
- Support encoder: YUV420/422
- Support decoder: YUV420/422/400
- Support 640M pixels/s, 10 fps continuous shoot

Display

- Support DSI v1.2 and D-PHY v1.2
- Support Dual MIPI
- Support QHD+@60fps/FHD+@120fps
- Support extend panel by DP V1.4
- Support 8 layer alpha-blending and writeback
- Support layer rotation
- Support color space conversion
- Support ProtectView:

- Support Blue Light Protection
- Support Sun Light Protection
- Support ViveView:
 - Support contrast enhancement
 - Support sharpness enhancement
 - Support HDR Playback
 - Support Wide Gamut Display, sRGB/DCI-P3
- Support GreenView:
 - Ambient Adaptive Lightness
 - Content Adaptive Brightness Control
 - Super Resolution

Camera

- Support four camera
 - Single Camera: 64M(4in1) 25fps ZSL capture, 108M(9in1) NonZSL capture
 - Dual Camera: 32M 25fps + 8M 25fps, 20M + 20M, 24M + 16M
 - Triple Camera: 64M(4in1) + 20M(UW) + 13M(T)
 - Four Camera: 64M(4in1) + 20M(UW) + 13M(T) + 2M
- Support four port MIPI CSI-2
 - MIPI CSI port0, 4-datalane, CPHY and DPHY combo, 3.5Gbps/Trio, 2.5Gbps/lane
 - MIPI CSI port1, 4-datalane, CPHY and DPHY combo, 2.5Gbps/Trio, 2.5Gbps/lane
 - MIPI CSI port2, 4-datalane or 2×2-datalane, 2.5Gbps/lane, support TX for log
 - MIPI CSI port3, 4-datalane or 2×2-datalane, 2.5Gbps/lane, support TX for log
- Support AI Face Detection
- Support dual PDAF
- Support TOF
- Support slow motion, up to 480fps@720p & 240fps@1080p, 60fps@4K
- Support super low light capturing for still and video
- Support auto scene recognition by SW or VDSP clarification:
 - Portrait
 - Children
 - Backlight & portrait
 - Night
 - Night & portrait
 - Low light
 - Document
 - Macro
 - Standard
 - Landscape
 - Black and white
 - Spotlight
 - Sports

- HDR
- Support Face recognition

Video Codec

- H.264
 - Compliance with ISO/IEC 14496-10-MPEG4 Part10, Advanced Video Coding & Scalable Video Coding
 - Encoder: progressive mode, YUV420, 8-bits
 - Decoder: progressive & interlaced mode, YUV420, 8-bits/10-bits
 - Encoder: baseline profile, main profile, high profile, level 5.1, 4k@24fps, 1080p@100fps, 720p@240fps
 - Decoder: baseline profile, main profile, high profile, high 10 profile, level 5.1, 4k@30fps
- H.265
 - Compliance with ISO/IEC 23008-2-Part2: High efficiency video coding
 - Encoder: Progressive mode, YUV420, 8-bits
 - Decoder: progressive mode, YUV420, 8-bits/10-bits
 - Encoder: main profile, level 5.1, 4k@60fps, 1080p@240fps
 - Decoder: main profile, main 10 profile, level 5.1, 4k@30fps 10bit, 4k@60fps 8bit
- VP8
 - Encoder to 1080p@30fps by software
 - Decoder to 1080p@60fps
- VP9
 - Encoder to 1080p@30fps by software
 - Decoder: profile0, profile2, 4k@30fps

FBC

- Support frame buffer compression
- Support AFBC1.2

Audio

- Support MP3/AAC/ADPCM/AMR/AMR_WB/FLAC decoding
- Support PCM record and playback
- Support Audio processing algorithm
- Support lower power MP3 decoding
- Support lower power A2DP
- Support USB audio
- Support low power Voice trigger

2

Package Information

2.1 Basic Information

Plastic-encapsulated surface mount packages are sensitive to damage induced by absorbed moisture and temperature. The chip is MSL 3, which is marked on the label of the package.

Table 2-1 Package Information

Device Name	UIS7885
Type	FCCSP
Body Size	12.5mm × 12.0mm
Height	0.869mm maximum
Ball count	852
Ball Pitch	0.4mm
Ball Size	0.26mm
Ball Matrix	43 × 41

2.2 Top Marking Definition

Figure 2-1 Silk screen printing

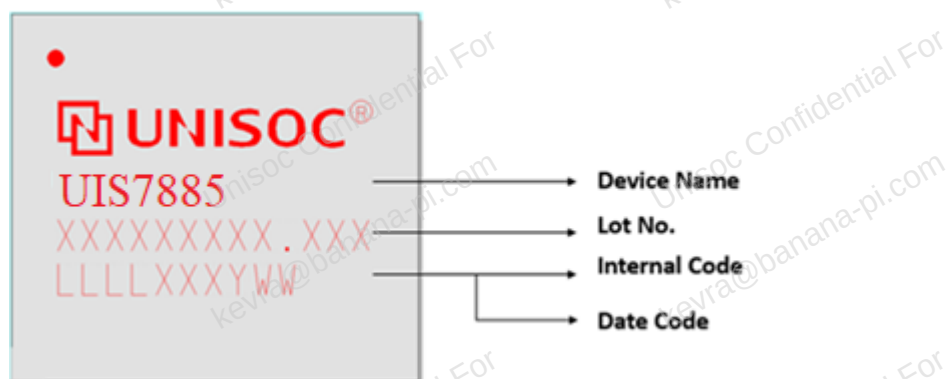


Figure 2-2 Package outline



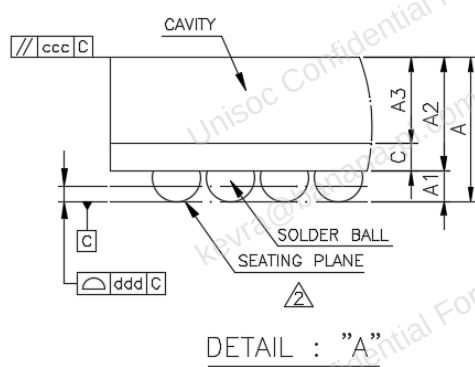
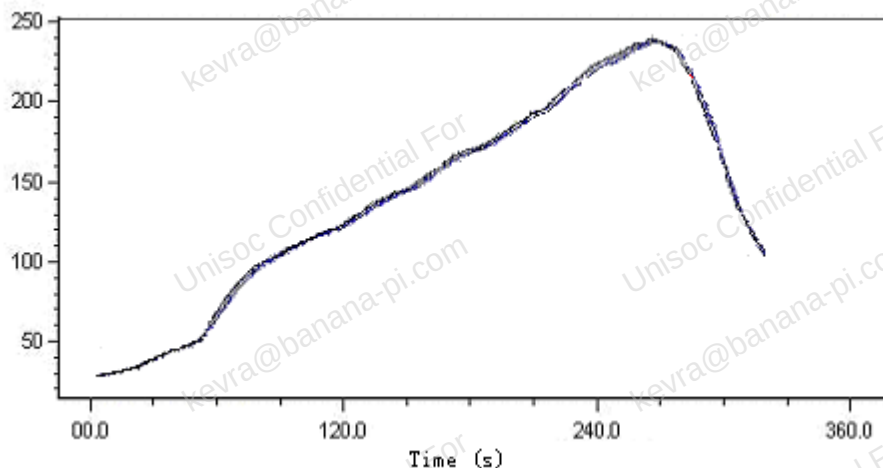


Figure 2-3 Package parameters

Symbol	Dimension in mm			Dimension in inch		
	MIN	NOM	MAX	MIN	NOM	MAX
A	0.7390	0.8040	0.8690	0.0291	0.0317	0.0342
A1	0.1200	0.1700	0.2200	0.0047	0.0067	0.0087
A2	0.5920	0.6340	0.6760	0.0233	0.0250	0.0266
A3	0.4200	0.4500	0.4800	0.0165	0.0177	0.0189
c	0.1540	0.1840	0.2140	0.0061	0.0072	0.0084
D	12.4000	12.5000	12.6000	0.4882	0.4921	0.4961
E	11.9000	12.0000	12.1000	0.4685	0.4724	0.4764
D1	——	11.8797	——	——	0.4677	——
E1	——	11.3140	——	——	0.4454	——
e	——	0.4000	——	——	0.0157	——
b	0.2100	0.2600	0.3100	0.0083	0.0102	0.0122
aaa	0.1000			0.0039		
ccc	0.2000			0.0079		
ddd	0.0800			0.0031		
eee	0.1500			0.0059		
fff	0.0500			0.0020		
MD/ME	43/41					
Note: refer to the SMT capacity for the package dimension of b.						

2.4 Reflow Profile

Figure 2-4 Reflow profile



NOTE

- Recommended reflow profile for lead-free solder paste
 - Ramp at 1-2°C per second to 245±5°C
 - Dwell at 235°C for 10s
 - Dwell at 217°C for 30~60s
 - Total reflow time is about 220~270s
 - Cold down ramp <4°C/s
- Recommended solder paste type
 - SnAgCu solder paste
 - Metal contents should be about 88.5%
- Recommended parameter for stencil making
 - Metal mask thickness: 5 mils
 - Opening area ratio: 100%

2.5 Device Moisture Sensitivity Level

Plastic-encapsulated surface mount packages are sensitive to damage induced by absorbed moisture and temperature. UIS7885 is MSL 3.

Table 2-2 MSL ratings summary

MSL	Out-of-bag floor life	Comments
1	Unlimited	<=30°C/85% RH
2	1 year	<=30°C/60% RH

MSL	Out-of-bag floor life	Comments
2a	4 weeks	$\leq 30^{\circ}\text{C}/60\% \text{ RH}$
3	168 hours	$\leq 30^{\circ}\text{C}/60\% \text{ RH}$
4	72 hours	$\leq 30^{\circ}\text{C}/60\% \text{ RH}$
5	48 hours	$\leq 30^{\circ}\text{C}/60\% \text{ RH}$
5a	24 hours	$\leq 30^{\circ}\text{C}/60\% \text{ RH}$
6	Mandatory bake before use. After bake, must be reflowed within the time limit specified on the label.	$\leq 30^{\circ}\text{C}/60\% \text{ RH}$

2.6 Thermal Characteristics

The thermal characteristics are as shown in [Table 2-3](#).

Table 2-3 Thermal characteristics

Symbol	Parameter	Condition	Value	Unit
Theta JA	Junction-to-Ambient thermal resistance	Air flow: 0 m/s	25.2	$^{\circ}\text{C}/\text{W}$

3.1 BGA Pin out

Figure 3-1 BGA Pin out



3.2 Pin Symbol Description

Table 3-1 Definition of pin symbols

Field	Symbol	Type Description
PinType	I	Digital input
	O	Digital output

Field	Symbol	Type Description
	O/T	Digital output with tri-state option
	I/O	Digital bi-directional pin
	I/O/T	Digital bi-directional pin with tri-state option
	PI	Power pin, input from external power supply
	PIO	Power pin, input from external or floating to use internal LDO power supply
	PO	Power pin, output for external devices
	G	Ground pin
	AI	Analog input pin
	AO	Analog output pin
	AIO	Analog bi-directional pin
Pin Value	IPU	Input with pull-up
	IPD	Input with pull-down
	OH	Output “1”
	OL	Output “0”
	Hiz	Tri-state
Power	VBAT	Battery power supply input
	VDDCORE	Power supply for core
	VDDCPU_LIT	Power supply for Little core CPU
	VDDCPU_MID	Power supply for Middle core CPU
	VDDCPU_BIG	Power supply for Big core CPU
	VDDGPU	Power supply for GPU
	VDDMM	Power supply for Multi-Media
	VDDMODEM	Power supply for Modem
	VDDPUB	Power supply for DDR controller
	VDDSRAM	Power supply for Dual-rail SRAM
	VDDSRAM_CPU_LIT	Power supply for Dual-rail SRAM of Little core CPU
	VDDSRAM_CPU_MID	Power supply for Dual-rail SRAM of Middle core CPU
	VDDAI	Power supply for NPU
	VSD0	Power supply for SDIO0 IO
	VSD2	Power supply for SDIO2 IO
	VEMMCIO	Power supply for eMMC IO

Field	Symbol	Type Description
	VSIM0 VSIM1	Power supply for SIM card 0/1
	VIO1V8	Power supply for I/O
	VIO1V8_ISE	ISE 1.8V digital power supply
	VDDMEM	Power supply for LPDDR4X core power
	VDDMEMQ	Power supply for LPDDR4X IO
	VDD0V75_PCIE	PCIe 0.75V digital power supply
	VDD0V75_UFS	UFS 0.75V digital power supply
	VDD0V75_USB	USB3.1 0.75V digital power supply
	AVDD1V2_DSI	DSI 1.2V analog power supply
	AVDD1V2_LVDSRF	LVDSRF 1.2V analog power supply
	AVDD1V2_CSI	CSI 1.2V analog power supply
	AVDD1V2_PCIE	PCIe 1.2V analog power supply
	AVDD1V2_UFS	UFS 1.2V analog power supply
	AVDD1V2_USB	USB3.1 1.2V analog power supply
	AVDD1V8_DPLL	DDR PLL 1.8V analog power supply
	AVDD1V8_DSI	DSI 1.8V analog power supply
	AVDD1V8_USB	USB2.0 1.8V analog power supply
	AVDD1V8_MPLLBIG	MPLLBIG 1.8V analog power supply
	AVDD1V8_MPLLLIT	MPLLLIT 1.8V analog power supply
	AVDD1V8_MPLLMID	MPLLMID 1.8V analog power supply
	AVDD1V8_PCIEPLL	PCIEPLL 1.8V analog power supply
	AVDD1V8_USBPLL	USB3.1 PLL 1.8V analog power supply
	AVDD1V8_EFU	EFUSE 1.8V analog power supply
	AVDD1V8_BB	Baseband DAC & PLL 1.8V analog power supply
	AVDD1V8_LVDSRF	LVDSRF 1.8V analog power supply
	AVDD1V8_CSI	CSI 1.8V analog power supply
	AVDD3V3_USB	USB2.0 3.3V analog power supply
Pin description	ie	Input enable, ie=0 means input disable, ie=1 means input enable
	oe	Output enable, oe=0 means output disable, oe=1, 1 means output enable, output high; oe=1, 0 means output enable, output low
	ie=0, oe=0	Hiz

Field	Symbol	Type Description
	wpu	Weak pull up resistor
	wpus	Weak pull up resistor selection
	wpd	Weak pull down resistor

3.3 Pin List

See *UIS7885 Pin List and Pin Register Instructions*.

3.4 Pin Description

Table 3-2 Pin description

Ball No	Ball Name	Description
LPDDR4X Interface		
N43	EMCA0_A/EMCA1_A	LPDDR4X channel A command or address 0/Discrete LPDDR4X channel A command or address 1
M42	EMCA1_A/EMCA0_A	LPDDR4X channel A command or address 1/Discrete LPDDR4X channel A command or address 0
K40	EMCA2_A/EMCA5_A	LPDDR4X channel A command or address 2/Discrete LPDDR4X channel A command or address 5
K42	EMCA3_A/EMCA3_A	LPDDR4X channel A command or address 3/Discrete LPDDR4X channel A command or address 3
L43	EMCA4_A/EMCA2_A	LPDDR4X channel A command or address 4/Discrete LPDDR4X channel A command or address 2
K38	EMCA5_A/EMCA4_A	LPDDR4X channel A command or address 5/Discrete LPDDR4X channel A command or address 4
M38	EMCK_C_A/EMCLK_A_N	LPDDR4X channel A differential clock negative/Discrete LPDDR4X channel A differential clock negative
N39	EMCK_T_A/EMCLK_A_P	LPDDR4X channel A differential clock positive/Discrete LPDDR4X channel A differential clock positive
L41	EMCKE0_A/EMCKE0_A	LPDDR4X channel A clock enable output 0/Discrete LPDDR4X channel A clock enable output 0
L39	EMCKE1_A/EMCKE1_A	LPDDR4X channel A clock enable output 1/Discrete LPDDR4X channel A clock enable output 1
M40	EMCS0_A/EMCS1_A	LPDDR4X channel A chip select 0/Discrete LPDDR4X channel A chip select 1

Ball No	Ball Name	Description
N41	EMCS1_A/EMCS0_A	LPDDR4X channel A chip select 1/Discrete LPDDR4X channel A chip select 0
H38	EMDMI0_A/EMDQ4_A	LPDDR4X channel A data mask Byte 0/Discrete LPDDR4X channel A data 4
E41	EMDQ0_A/EMDQ6_A	LPDDR4X channel A data 0/Discrete LPDDR4X channel A data 6
F40	EMDQ1_A/EMDQ7_A	LPDDR4X channel A data 1/Discrete LPDDR4X channel A data 7
G41	EMDQ2_A/EMDMI0_A	LPDDR4X channel A data 2/Discrete LPDDR4X channel A data mask Byte 0
H40	EMDQ3_A/EMDQ5_A	LPDDR4X channel A data 3/Discrete LPDDR4X channel A data 5
E43	EMDQ4_A/EMDQ0_A	LPDDR4X channel A data 4/Discrete LPDDR4X channel A data 0
G43	EMDQ5_A/EMDQ2_A	LPDDR4X channel A data 5/Discrete LPDDR4X channel A data 2
H42	EMDQ6_A/EMDQ3_A	LPDDR4X channel A data 6/Discrete LPDDR4X channel A data 3
F42	EMDQ7_A/EMDQ1_A	LPDDR4X channel A data 7/Discrete LPDDR4X channel A data 1
G39	EMDQS0_C_A/EMDQS0_A_N	LPDDR4X channel A data strobe 0 differential negative/Discrete LPDDR4X channel A data strobe 0 differential negative
F38	EMDQS0_T_A/EMDQS0_A_P	LPDDR4X channel A data strobe 0 differential positive/Discrete LPDDR4X channel A data strobe 0 differential positive
T40	EMDMI1_A/EMDQ11_A	LPDDR4X channel A data mask Byte 1/Discrete LPDDR4X channel A data 11
V38	EMDQ8_A/EMDQ8_A	LPDDR4X channel A data 8/Discrete LPDDR4X channel A data 8
V42	EMDQ9_A/EMDQ12_A	LPDDR4X channel A data 9/Discrete LPDDR4X channel A data 12
V40	EMDQ10_A/EMDQ9_A	LPDDR4X channel A data 10/Discrete LPDDR4X channel A data 9
U41	EMDQ11_A/EMDQ10_A	LPDDR4X channel A data 11/Discrete LPDDR4X channel A data 10
U43	EMDQ12_A/EMDQ13_A	LPDDR4X channel A data 12/Discrete LPDDR4X channel A data 13
R41	EMDQ13_A/EMDMI1_A	LPDDR4X channel A data 13/Discrete LPDDR4X channel A data mask Byte 1

Ball No	Ball Name	Description
T42	EMDQ14_A/EMDQ14_A	LPDDR4X channel A data 14/Discrete LPDDR4X channel A data 14
R43	EMDQ15_A/EMDQ15_A	LPDDR4X channel A data 15/Discrete LPDDR4X channel A data 15
T38	EMDQS1_C_A/EMDQS1_A_N	LPDDR4X channel A data strobe 1 differential negative/Discrete LPDDR4X channel A data strobe 1 differential negative
U39	EMDQS1_T_A/EMDQS1_A_P	LPDDR4X channel A data strobe 1 differential positive/Discrete LPDDR4X channel A data strobe 1 differential positive
AE43	EMCA0_B/EMCA1_B	LPDDR4X channel B command or address 0/Discrete LPDDR4X channel B command or address 1
AF42	EMCA1_B/EMCA0_B	LPDDR4X channel B command or address 1/Discrete LPDDR4X channel B command or address 0
AH40	EMCA2_B/EMCA5_B	LPDDR4X channel B command or address 2/Discrete LPDDR4X channel B command or address 5
AH42	EMCA3_B/EMCA3_B	LPDDR4X channel B command or address 3/Discrete LPDDR4X channel B command or address 3
AG43	EMCA4_B/EMCA2_B	LPDDR4X channel B command or address 4/Discrete LPDDR4X channel B command or address 2
AH38	EMCA5_B/EMCA4_B	LPDDR4X channel B command or address 5/Discrete LPDDR4X channel B command or address 4
AF38	EMCK_C_B/EMCLK_B_N	LPDDR4X channel B differential clock negative/Discrete LPDDR4X channel B differential clock negative
AE39	EMCK_T_B/EMCLK_B_P	LPDDR4X channel B differential clock positive/Discrete LPDDR4X channel B differential clock positive
AG41	EMCKE0_B/EMCKE0_B	LPDDR4X channel B clock enable output 0/Discrete LPDDR4X channel B clock enable output 0
AG39	EMCKE1_B/EMCKE1_B	LPDDR4X channel B clock enable output 1/Discrete LPDDR4X channel B clock enable output 1
AF40	EMCS0_B/EMCS1_B	LPDDR4X channel B chip select 0/Discrete LPDDR4X channel B chip select 1
AE41	EMCS1_B/EMCS0_B	LPDDR4X channel B chip select 1/Discrete LPDDR4X channel B chip select 0
AK38	EMDMI0_B/EMDQ4_B	LPDDR4X channel B data mask Byte 0/Discrete LPDDR4X channel B data 4
AN41	EMDQ0_B/EMDQ6_B	LPDDR4X channel B data 0/Discrete LPDDR4X channel B data 6
AM40	EMDQ1_B/EMDQ7_B	LPDDR4X channel B data 1/Discrete LPDDR4X channel B data 7

Ball No	Ball Name	Description
AL41	EMDQ2_B/EMDMI0_B	LPDDR4X channel B data 2/Discrete LPDDR4X channel B data mask Byte 0
AK40	EMDQ3_B/EMDQ5_B	LPDDR4X channel B data 3/Discrete LPDDR4X channel B data 5
AN43	EMDQ4_B/EMDQ0_B	LPDDR4X channel B data 4/Discrete LPDDR4X channel B data 0
AL43	EMDQ5_B/EMDQ2_B	LPDDR4X channel B data 5/Discrete LPDDR4X channel B data 2
AK42	EMDQ6_B/EMDQ3_B	LPDDR4X channel B data 6/Discrete LPDDR4X channel B data 3
AM42	EMDQ7_B/EMDQ1_B	LPDDR4X channel B data 7/Discrete LPDDR4X channel B data 1
AL39	EMDQS0_C_B/EMDQS0_B_N	LPDDR4X channel A data strobe 0 differential negative/Discrete LPDDR4X channel B data strobe 0 differential negative
AM38	EMDQS0_T_B/EMDQS0_B_P	LPDDR4X channel A data strobe 0 differential positive/Discrete LPDDR4X channel B data strobe 0 differential positive
AB40	EMDMI1_B/EMDQ11_B	LPDDR4X channel B data mask Byte 1/Discrete LPDDR4X channel B data 11
Y38	EMDQ8_B/EMDQ8_B	LPDDR4X channel B data 8/Discrete LPDDR4X channel B data 8
Y42	EMDQ9_B/EMDQ12_B	LPDDR4X channel B data 9/Discrete LPDDR4X channel B data 12
Y40	EMDQ10_B/EMDQ9_B	LPDDR4X channel B data 10/Discrete LPDDR4X channel B data 9
AA41	EMDQ11_B/EMDQ10_B	LPDDR4X channel B data 11/Discrete LPDDR4X channel B data 10
AA43	EMDQ12_B/EMDQ13_B	LPDDR4X channel B data 12/Discrete LPDDR4X channel B data 13
AC41	EMDQ13_B/EMDMI1_B	LPDDR4X channel B data 13/Discrete LPDDR4X channel B data mask Byte 1
AB42	EMDQ14_B/EMDQ14_B	LPDDR4X channel B data 14/Discrete LPDDR4X channel B data 14
AC43	EMDQ15_B/EMDQ15_B	LPDDR4X channel B data 15/Discrete LPDDR4X channel B data 15
AB38	EMDQS1_C_B/EMDQS1_B_N	LPDDR4X channel A data strobe 1 differential negative/Discrete LPDDR4X channel B data strobe 1 differential negative

Ball No	Ball Name	Description
AA39	EMDQS1_T_B/EMDQS1_B_P	LPDDR4X channel A data strobe 1 differential positive/Discrete LPDDR4X channel B data strobe 1 differential positive
W43	EMRSTN	LPDDR4X reset , low active
C43	ZQ	LPDDR4X Impedance calibration
Thermal Test Interface		
AU33	SSUSB_CKN	USB3.1 backup clock input negative
AW33	SSUSB_CKP	USB3.1 backup clock input positive
AT38	UFS_CKN	UFS backup clock input negative
AV38	UFS_CKP	UFS backup clock input positive
V10	CLKOUT_TEST_PLL	PLL clock test point
H26	CLKTEST_MPLLBIG	MPLLBIG clock test point
Y22	CLKTEST_MPLLLIT	MPLLLIT clock test point
Y8	TEST_THM_DP	Thermal Test Diode anode
F32	TSET_DDC_LS	DDC low speed test point
CSI Interface		
C5	CSI0_CKN_C1	CSI0 D-PHY differential clock 0 negative/CSI0 C-PHY lane 1 C
E5	CSI0_CKP_B1	CSI0 D-PHY differential clock 0 positive/CSI0 C-PHY lane 1 B
B2	CSI0_DN0_B0	CSI0 D-PHY differential data 0 negative/CSI0 C-PHY lane 0 B
A3	CSI0_DP0_A0	CSI0 D-PHY differential data 0 positive/CSI0 C-PHY lane 0 A
D4	CSI0_DN1_A1	CSI0 D-PHY differential data 1 negative/CSI0 C-PHY lane 1 A
C3	CSI0_DP1_C0	CSI0 D-PHY differential data 1 positive/CSI0 C-PHY lane 0 C
D8	CSI0_DN2_B2	CSI0 D-PHY differential data 2 negative/CSI0 C-PHY lane 2 B
F8	CSI0_DP2_A2	CSI0 D-PHY differential data 2 positive/CSI0 C-PHY lane 2 A
E7	CSI0_DN3_NC	CSI0 D-PHY differential data 3 negative/Not Connect
C7	CSI0_DP3_C2	CSI0 D-PHY differential data 3 positive/CSI0 C-PHY lane 2 C
H2	CSI1_CKN_C1	CSI1 D-PHY differential clock 0 negative/CSI1 C-PHY lane 1 C
G1	CSI1_CKP_B1	CSI1 D-PHY differential clock 0 positive/CSI1 C-PHY lane 1 B
F4	CSI1_DN0_B0	CSI1 D-PHY differential data 0 negative/CSI1 C-PHY lane 0 B
F6	CSI1_DP0_A0	CSI1 D-PHY differential data 0 positive/CSI1 C-PHY lane 0 A

Ball No	Ball Name	Description
G3	CSI1_DN1_A1	CSI1 D-PHY differential data 1 negative/CSI1 C-PHY lane 1 A
G5	CSI1_DP1_C0	CSI1 D-PHY differential data 1 positive/CSI1 C-PHY lane 0 C
E1	CSI1_DN2_B2	CSI1 D-PHY differential data 2 negative/CSI1 C-PHY lane 2 B
F2	CSI1_DP2_A2	CSI1 D-PHY differential data 2 positive/CSI1 C-PHY lane 2 A
C1	CSI1_DN3_NC	CSI1 D-PHY differential data 3 negative/Not Connect
D2	CSI1_DP3_C2	CSI1 D-PHY differential data 3 positive/CSI1 C-PHY lane 2 C
A7	CSI2_CKN0	CSI2 differential clock 0 negative
B6	CSI2_CKP0	CSI2 differential clock 0 positive
A5	CSI2_CKN1	CSI2 differential clock 1 negative
B4	CSI2_CKP1	CSI2 differential clock 1 positive
A11	CSI2_DN0	CSI2 differential data 0 negative
B10	CSI2_DP0	CSI2 differential data 0 positive
A9	CSI2_DN1	CSI2 differential data 1 negative
B8	CSI2_DP1	CSI2 differential data 1 positive
C9	CSI2_DN2	CSI2 differential data 2 negative
E9	CSI2_DP2	CSI2 differential data 2 positive
D10	CSI2_DN3	CSI2 differential data 3 negative
F10	CSI2_DP3	CSI2 differential data 3 positive
C13	CSI3_CKN0	CSI3 differential clock 0 negative
E13	CSI3_CKP0	CSI3 differential clock 0 positive
F14	CSI3_CKN1	CSI3 differential clock 1 negative
D14	CSI3_CKP1	CSI3 differential clock 1 positive
D12	CSI3_DN0	CSI3 differential data 0 negative
F12	CSI3_DP0	CSI3 differential data 0 positive
A13	CSI3_DN1	CSI3 differential data 1 negative
B12	CSI3_DP1	CSI3 differential data 1 positive
B14	CSI3_DN2	CSI3 differential data 2 negative
A15	CSI3_DP2	CSI3 differential data 2 positive
C15	CSI3_DN3	CSI3 differential data 3 negative
E15	CSI3_DP3	CSI3 differential data 3 positive
Clock Interface		

Ball No	Ball Name	Description
Y2	CLK26M_OUT_PMIC	26M clock output for PMIC
Y4	CLK52M_IN	52M clock input for AP
F36	CLK52M_IN_PCIE	52M clock input for PCIE
AT34	CLK52M_IN_USB	52M clock input for USB and UFS
DSI Interface		
AT28	DSI0_CKN	DSI0 differential clock negative
AV28	DSI0_CKP	DSI0 differential clock positive
AY28	DSI0_DN0	DSI0 differential data 0 negative
BA29	DSI0_DP0	DSI0 differential data 0 positive
AW29	DSI0_DN1	DSI0 differential data 1 negative
AU29	DSI0_DP1	DSI0 differential data 1 positive
AW27	DSI0_DN2	DSI0 differential data 2 negative
AU27	DSI0_DP2	DSI0 differential data 2 positive
AY26	DSI0_DN3	DSI0 differential data 3 negative
BA27	DSI0_DP3	DSI0 differential data 3 positive
AT26	DSI1_CKN	DSI1 differential clock negative
AP26	DSI1_CKP	DSI1 differential clock positive
AW23	DSI1_DN0	DSI1 differential data 0 negative
AU23	DSI1_DP0	DSI1 differential data 0 positive
AT24	DSI1_DN1	DSI1 differential data 1 negative
AV24	DSI1_DP1	DSI1 differential data 1 positive
AU25	DSI1_DN2	DSI1 differential data 2 negative
AW25	DSI1_DP2	DSI1 differential data 2 positive
BA25	DSI1_DN3	DSI1 differential data 3 negative
AY24	DSI1_DP3	DSI1 differential data 3 positive
LVDSRF Analog Interface		
AC5	LVDSRF_DL0_N	LVDSRF Downlink data 0 negative
AC3	LVDSRF_DL0_P	LVDSRF Downlink data 0 positive
AB6	LVDSRF_DL1_N	LVDSRF Downlink data 1 negative
AB4	LVDSRF_DL1_P	LVDSRF Downlink data 1 positive
AA1	LVDSRF_DL2_N	LVDSRF Downlink data 2 negative

Ball No	Ball Name	Description
AB2	LVDSRF_DL2_P	LVDSRF Downlink data 2 positive
AC1	LVDSRF_DL3_N	LVDSRF Downlink data 3 negative
AD2	LVDSRF_DL3_P	LVDSRF Downlink data 3 positive
AF6	LVDSRF_UL0_N	LVDSRF Uplink data 0 negative
AF4	LVDSRF_UL0_P	LVDSRF Uplink data 0 positive
AE1	LVDSRF_UL1_N	LVDSRF Uplink data 1 negative
AF2	LVDSRF_UL1_P	LVDSRF Uplink data 1 positive
AE3	LVDSRF_UL2_N	LVDSRF Uplink data 2 negative
AE5	LVDSRF_UL2_P	LVDSRF Uplink data 2 positive
AA7	LVDSRF_TEST_HS_ON	LVDSRF highspeed negative test ball
AB8	LVDSRF_TEST_HS_OP	LVDSRF highspeed positive test ball
AC7	LVDSRF_TEST_LS	LVDSRF lowspeed test ball
APC Interface		
W1	APCOUT0	APC output 0
W5	APCOUT1	APC output 1
USB2.0 Interface		
AR31	USB_DM	USB2.0 data negative
AP30	USB_DP	USB2.0 data positive
AY30	USB_REXT	USB2.0 calibration pin
USB3.1 Interface		
AT32	SSUSB_RXN1	USB3.1 RX 1 negative
AV32	SSUSB_RXP1	USB3.1 RX 1 positive
AY32	SSUSB_RXN2	USB3.1 RX 2 negative
BA33	SSUSB_RXP2	USB3.1 RX 2 positive
BA35	SSUSB_TXN1	USB3.1 TX 1 negative
AY36	SSUSB_TXP1	USB3.1 TX 1 positive
AW35	SSUSB_TXN2	USB3.1 TX 2 negative
AU35	SSUSB_TXP2	USB3.1 TX 2 positive
BA31	SSUSB_REXT	USB3.1 calibration pin
AW31	USB_DP_AUX1	USB DP AUX channel 1
AU31	USB_DP_AUX2	USB DP AUX channel 2

Ball No	Ball Name	Description
UFS Interface		
AR35	UFS_CLK	UFS reference clock output
AP34	UFS_RST	UFS reset output
AU37	UFS_RXN0	UFS RX0 negative
AW37	UFS_RXP0	UFS RX0 positive
AU39	UFS_RXN1	UFS RX1 negative
AW39	UFS_RXP1	UFS RX1 positive
BA41	UFS_TXN0	UFS TX0 negative
AY40	UFS_TXP0	UFS TX0 positive
BA39	UFS_TXN1	UFS TX1 negative
AY38	UFS_TXP1	UFS TX1 positive
AR37	UFS_REXT	UFS calibration pin
PCIe Interface		
D34	PCIE_CKN	PCIe differential clock negative
F34	PCIE_CKP	PCIe differential clock positive
C33	PCIE_RXN	PCIe RX negative
E33	PCIE_RXP	PCIe RX positive
D36	PCIE_TXN	PCIe TX negative
E35	PCIE_TXP	PCIe TX positive
K34	PCIE_REXT	PCIe calibration pin
System Power Input		
J9	AVDD1V2_CSI	CSI 1.2V analog power supply
J11	AVDD1V2_CSI	CSI 1.2V analog power supply
AM30	AVDD1V2_DSI	DSI 1.2V analog power supply
AC9	AVDD1V2_LVDSRF	LVDSRF 1.2V analog power supply
H32	AVDD1V2_PCIE	PCIe 1.2V analog power supply
AN35	AVDD1V2_UFS	UFS 1.2V analog power supply
AM32	AVDD1V2_USB	USB3.1 1.2V analog power supply
W9	AVDD1V8_BB	Baseband DAC & PLL 1.8V analog power supply
J13	AVDD1V8_CSI	CSI 1.8V analog power supply
W35	AVDD1V8_DPLL	DDR PLL 1.8V analog power supply

Ball No	Ball Name	Description
AP28	AVDD1V8_DSI	DSI 1.8V analog power supply
H36	AVDD1V8_EFU	EFUSE 1.8V analog power supply
AA9	AVDD1V8_LVDSRF	LVDSRF 1.8V analog power supply
H30	AVDD1V8_MPLLBIG	MPLLBIG 1.8V analog power supply
V14	AVDD1V8_MPLLLIT	MPLLLIT 1.8V analog power supply
J23	AVDD1V8_MPLLMID	MPLLMID 1.8V analog power supply
H34	AVDD1V8_PCIEPLL	PCIEPLL 1.8V analog power supply
AN29	AVDD1V8_USB	USB2.0 1.8V analog power supply
AM34	AVDD1V8_USBPLL	USB3.1 PLL 1.8V analog power supply
AN31	AVDD3V3_USB	USB2.0 3.3V analog power supply
J33	VDD0V75_PCIE	PCIe 0.75V digital power supply
AL35	VDD0V75_UFS	UFS 0.75V digital power supply
AL33	VDD0V75_USB	USB3.1 0.75V digital power supply
L33	VDD1V8_ISE	ISE 1.8V digital power supply
AK36	VEMMCIO	Power supply for eMMC IO
F30	VIO1V8	Power supply for digital IO
G25	VIO1V8	Power supply for digital IO
G31	VIO1V8	Power supply for digital IO
J37	VIO1V8	Power supply for digital IO
T8	VIO1V8	Power supply for digital IO
AF8	VIO1V8	Power supply for digital IO
AP18	VIO1V8	Power supply for digital IO
AT14	VSD0	Power supply for SDIO0 IO
AR13	VSD2	Power supply for SDIO2 IO
AR15	VSIM0	Power supply for SIM card 0
AP14	VSIM1	Power supply for SIM card 1
K8	VDDAI	Power supply for AI
K10	VDDAI	Power supply for AI
L9	VDDAI	Power supply for AI
M8	VDDAI	Power supply for AI
N9	VDDAI	Power supply for AI

Ball No	Ball Name	Description
R9	VDDAI	Power supply for AI
H24	VDDCORE	Power supply for Core domain
J19	VDDCORE	Power supply for Core domain
L35	VDDCORE	Power supply for Core domain
M34	VDDCORE	Power supply for Core domain
N33	VDDCORE	Power supply for Core domain
U9	VDDCORE	Power supply for Core domain
U11	VDDCORE	Power supply for Core domain
U19	VDDCORE	Power supply for Core domain
V30	VDDCORE	Power supply for Core domain
W17	VDDCORE	Power supply for Core domain
W19	VDDCORE	Power supply for Core domain
W21	VDDCORE	Power supply for Core domain
W23	VDDCORE	Power supply for Core domain
W25	VDDCORE	Power supply for Core domain
W27	VDDCORE	Power supply for Core domain
W29	VDDCORE	Power supply for Core domain
AA23	VDDCORE	Power supply for Core domain
AB22	VDDCORE	Power supply for Core domain
AC21	VDDCORE	Power supply for Core domain
AD8	VDDCORE	Power supply for Core domain
AD22	VDDCORE	Power supply for Core domain
AE23	VDDCORE	Power supply for Core domain
AG23	VDDCORE	Power supply for Core domain
AH24	VDDCORE	Power supply for Core domain
AH30	VDDCORE	Power supply for Core domain
AH32	VDDCORE	Power supply for Core domain
AJ33	VDDCORE	Power supply for Core domain
AN17	VDDCORE	Power supply for Core domain
AN27	VDDCORE	Power supply for Core domain
J29	VDDCPU_BIG	Power supply for Big core CPU

Ball No	Ball Name	Description
K30	VDDCPU_BIG	Power supply for Big core CPU
L29	VDDCPU_BIG	Power supply for Big core CPU
M30	VDDCPU_BIG	Power supply for Big core CPU
N29	VDDCPU_BIG	Power supply for Big core CPU
P20	VDDCPU_LIT	Power supply for Little core CPU
P28	VDDCPU_LIT	Power supply for Little core CPU
R21	VDDCPU_LIT	Power supply for Little core CPU
R25	VDDCPU_LIT	Power supply for Little core CPU
R29	VDDCPU_LIT	Power supply for Little core CPU
T24	VDDCPU_LIT	Power supply for Little core CPU
U21	VDDCPU_LIT	Power supply for Little core CPU
U25	VDDCPU_LIT	Power supply for Little core CPU
U29	VDDCPU_LIT	Power supply for Little core CPU
J21	VDDCPU_MID	Power supply for Middle core CPU
J25	VDDCPU_MID	Power supply for Middle core CPU
K22	VDDCPU_MID	Power supply for Middle core CPU
K26	VDDCPU_MID	Power supply for Middle core CPU
L21	VDDCPU_MID	Power supply for Middle core CPU
L25	VDDCPU_MID	Power supply for Middle core CPU
M22	VDDCPU_MID	Power supply for Middle core CPU
M26	VDDCPU_MID	Power supply for Middle core CPU
Y32	VDDGPU	Power supply for GPU
AA27	VDDGPU	Power supply for GPU
AA31	VDDGPU	Power supply for GPU
AB26	VDDGPU	Power supply for GPU
AB32	VDDGPU	Power supply for GPU
AC25	VDDGPU	Power supply for GPU
AC33	VDDGPU	Power supply for GPU
AD26	VDDGPU	Power supply for GPU
AD32	VDDGPU	Power supply for GPU
AE27	VDDGPU	Power supply for GPU

Ball No	Ball Name	Description
AE31	VDDGPU	Power supply for GPU
AF26	VDDGPU	Power supply for GPU
AF28	VDDGPU	Power supply for GPU
AF32	VDDGPU	Power supply for GPU
AG25	VDDGPU	Power supply for GPU
AG33	VDDGPU	Power supply for GPU
T36	VDDMEM	Power supply for LPDDR4X core
AD36	VDDMEM	Power supply for LPDDR4X core
M36	VDDMEMQ	Power supply for LPDDR4X IO
N35	VDDMEMQ	Power supply for LPDDR4X IO
P36	VDDMEMQ	Power supply for LPDDR4X IO
R35	VDDMEMQ	Power supply for LPDDR4X IO
U35	VDDMEMQ	Power supply for LPDDR4X IO
Y36	VDDMEMQ	Power supply for LPDDR4X IO
AA35	VDDMEMQ	Power supply for LPDDR4X IO
AB36	VDDMEMQ	Power supply for LPDDR4X IO
AC35	VDDMEMQ	Power supply for LPDDR4X IO
AE35	VDDMEMQ	Power supply for LPDDR4X IO
K12	VDDMM	Power supply for Multi-Media
K18	VDDMM	Power supply for Multi-Media
L13	VDDMM	Power supply for Multi-Media
L17	VDDMM	Power supply for Multi-Media
M14	VDDMM	Power supply for Multi-Media
M16	VDDMM	Power supply for Multi-Media
N13	VDDMM	Power supply for Multi-Media
N17	VDDMM	Power supply for Multi-Media
P14	VDDMM	Power supply for Multi-Media
P18	VDDMM	Power supply for Multi-Media
R17	VDDMM	Power supply for Multi-Media
T16	VDDMM	Power supply for Multi-Media
U15	VDDMM	Power supply for Multi-Media

Ball No	Ball Name	Description
U17	VDDMM	Power supply for Multi-Media
AJ23	VDDMM	Power supply for Multi-Media
AJ25	VDDMM	Power supply for Multi-Media
AJ29	VDDMM	Power supply for Multi-Media
AK24	VDDMM	Power supply for Multi-Media
AK30	VDDMM	Power supply for Multi-Media
AL23	VDDMM	Power supply for Multi-Media
AL29	VDDMM	Power supply for Multi-Media
AM22	VDDMM	Power supply for Multi-Media
AM24	VDDMM	Power supply for Multi-Media
W11	VDDMODEM	Power supply for Modem
Y12	VDDMODEM	Power supply for Modem
AA11	VDDMODEM	Power supply for Modem
AA17	VDDMODEM	Power supply for Modem
AB16	VDDMODEM	Power supply for Modem
AC11	VDDMODEM	Power supply for Modem
AC17	VDDMODEM	Power supply for Modem
AD10	VDDMODEM	Power supply for Modem
AD16	VDDMODEM	Power supply for Modem
AE11	VDDMODEM	Power supply for Modem
AE15	VDDMODEM	Power supply for Modem
AE21	VDDMODEM	Power supply for Modem
AF10	VDDMODEM	Power supply for Modem
AF14	VDDMODEM	Power supply for Modem
AF20	VDDMODEM	Power supply for Modem
AG19	VDDMODEM	Power supply for Modem
AH10	VDDMODEM	Power supply for Modem
AH14	VDDMODEM	Power supply for Modem
AH20	VDDMODEM	Power supply for Modem
AJ11	VDDMODEM	Power supply for Modem
AJ15	VDDMODEM	Power supply for Modem

Ball No	Ball Name	Description
AJ19	VDDMODEM	Power supply for Modem
AK10	VDDMODEM	Power supply for Modem
AK14	VDDMODEM	Power supply for Modem
AK20	VDDMODEM	Power supply for Modem
AL9	VDDMODEM	Power supply for Modem
AL15	VDDMODEM	Power supply for Modem
AL19	VDDMODEM	Power supply for Modem
AM10	VDDMODEM	Power supply for Modem
AM14	VDDMODEM	Power supply for Modem
AN9	VDDMODEM	Power supply for Modem
AN15	VDDMODEM	Power supply for Modem
P34	VDDPUB	Power supply for DDR controller
T34	VDDPUB	Power supply for DDR controller
V34	VDDPUB	Power supply for DDR controller
Y34	VDDPUB	Power supply for DDR controller
AB34	VDDPUB	Power supply for DDR controller
AD34	VDDPUB	Power supply for DDR controller
AF34	VDDPUB	Power supply for DDR controller
J17	VDDSRAM	Power supply for Dual-rail SRAM
K16	VDDSRAM	Power supply for Dual-rail SRAM
M12	VDDSRAM	Power supply for Dual-rail SRAM
T14	VDDSRAM	Power supply for Dual-rail SRAM
W31	VDDSRAM	Power supply for Dual-rail SRAM
Y14	VDDSRAM	Power supply for Dual-rail SRAM
Y18	VDDSRAM	Power supply for Dual-rail SRAM
Y30	VDDSRAM	Power supply for Dual-rail SRAM
AA13	VDDSRAM	Power supply for Dual-rail SRAM
AA19	VDDSRAM	Power supply for Dual-rail SRAM
AB14	VDDSRAM	Power supply for Dual-rail SRAM
AB18	VDDSRAM	Power supply for Dual-rail SRAM
AB30	VDDSRAM	Power supply for Dual-rail SRAM

Ball No	Ball Name	Description
AC29	VDDSRAM	Power supply for Dual-rail SRAM
AD14	VDDSRAM	Power supply for Dual-rail SRAM
AD18	VDDSRAM	Power supply for Dual-rail SRAM
AE13	VDDSRAM	Power supply for Dual-rail SRAM
AE19	VDDSRAM	Power supply for Dual-rail SRAM
AE29	VDDSRAM	Power supply for Dual-rail SRAM
AF18	VDDSRAM	Power supply for Dual-rail SRAM
AG13	VDDSRAM	Power supply for Dual-rail SRAM
AG17	VDDSRAM	Power supply for Dual-rail SRAM
AH18	VDDSRAM	Power supply for Dual-rail SRAM
AJ13	VDDSRAM	Power supply for Dual-rail SRAM
AK18	VDDSRAM	Power supply for Dual-rail SRAM
AK26	VDDSRAM	Power supply for Dual-rail SRAM
AL13	VDDSRAM	Power supply for Dual-rail SRAM
AL27	VDDSRAM	Power supply for Dual-rail SRAM
AM12	VDDSRAM	Power supply for Dual-rail SRAM
AM18	VDDSRAM	Power supply for Dual-rail SRAM
AM26	VDDSRAM	Power supply for Dual-rail SRAM
P26	VDDSRAM_CPU_LIT	Power supply for Dual-rail SRAM of Little core CPU
T26	VDDSRAM_CPU_LIT	Power supply for Dual-rail SRAM of Little core CPU
L23	VDDSRAM_CPU_MID	Power supply for Dual-rail SRAM of Middle core CPU
Ground		
W3	AVSS_BB	Analog ground of Baseband DAC and PLL
B16	AVSS_CSI	Analog ground of CSI
C11	AVSS_CSI	Analog ground of CSI
D6	AVSS_CSI	Analog ground of CSI
D16	AVSS_CSI	Analog ground of CSI
E3	AVSS_CSI	Analog ground of CSI
E11	AVSS_CSI	Analog ground of CSI
G7	AVSS_CSI	Analog ground of CSI
G9	AVSS_CSI	Analog ground of CSI

Ball No	Ball Name	Description
G11	AVSS_CSI	Analog ground of CSI
G13	AVSS_CSI	Analog ground of CSI
H4	AVSS_CSI	Analog ground of CSI
J1	AVSS_CSI	Analog ground of CSI
G33	AVSS_DDC	Analog ground of DDC
V36	AVSS_DPLL	Analog ground of DDR PLL
AR25	AVSS_DSI	Analog ground of DSI
AR27	AVSS_DSI	Analog ground of DSI
AR29	AVSS_DSI	Analog ground of DSI
AV26	AVSS_DSI	Analog ground of DSI
BA23	AVSS_DSI	Analog ground of DSI
R33	AVSS_ISE	Analog ground of ISE
AA3	AVSS_LVDSRF	Analog ground of LVDSRF
AA5	AVSS_LVDSRF	Analog ground of LVDSRF
AD4	AVSS_LVDSRF	Analog ground of LVDSRF
AD6	AVSS_LVDSRF	Analog ground of LVDSRF
AE7	AVSS_LVDSRF	Analog ground of LVDSRF
AG1	AVSS_LVDSRF	Analog ground of LVDSRF
AG5	AVSS_LVDSRF	Analog ground of LVDSRF
H28	AVSS_MPLLBIG	Analog ground of MPLLBIG
Y20	AVSS_MPLLLIT	Analog ground of MPLLLIT
H22	AVSS_MPLLMID	Analog ground of MPLLMID
E37	AVSS_PCIEPLL	Analog ground of PCIEPLL
G35	AVSS_PCIEPLL	Analog ground of PCIEPLL
J35	AVSS_RCO	Analog ground of RCO
T10	AVSS_THM	Analog ground of Thermal sensor
AR33	AVSS_USBPLL	Analog ground of USBPLL
AT36	AVSS_USBPLL	Analog ground of USBPLL
C35	VSS	Ground
D24	VSS	Ground
D32	VSS	Ground

Ball No	Ball Name	Description
D40	VSS	Ground
D42	VSS	Ground
E39	VSS	Ground
G37	VSS	Ground
J15	VSS	Ground
J27	VSS	Ground
J31	VSS	Ground
J39	VSS	Ground
J41	VSS	Ground
J43	VSS	Ground
K14	VSS	Ground
K20	VSS	Ground
K24	VSS	Ground
K28	VSS	Ground
K32	VSS	Ground
K36	VSS	Ground
L7	VSS	Ground
L11	VSS	Ground
L15	VSS	Ground
L19	VSS	Ground
L27	VSS	Ground
L31	VSS	Ground
L37	VSS	Ground
M10	VSS	Ground
M18	VSS	Ground
M20	VSS	Ground
M28	VSS	Ground
M32	VSS	Ground
N7	VSS	Ground
N15	VSS	Ground
N19	VSS	Ground

Ball No	Ball Name	Description
N21	VSS	Ground
N27	VSS	Ground
P8	VSS	Ground
P16	VSS	Ground
P38	VSS	Ground
P40	VSS	Ground
P42	VSS	Ground
R15	VSS	Ground
R19	VSS	Ground
R27	VSS	Ground
R37	VSS	Ground
R39	VSS	Ground
T18	VSS	Ground
T20	VSS	Ground
T22	VSS	Ground
T28	VSS	Ground
U13	VSS	Ground
U23	VSS	Ground
U27	VSS	Ground
U31	VSS	Ground
U33	VSS	Ground
U37	VSS	Ground
V12	VSS	Ground
V16	VSS	Ground
V18	VSS	Ground
V20	VSS	Ground
V22	VSS	Ground
V24	VSS	Ground
V26	VSS	Ground
V28	VSS	Ground
V32	VSS	Ground

Ball No	Ball Name	Description
W13	VSS	Ground
W15	VSS	Ground
W37	VSS	Ground
W39	VSS	Ground
W41	VSS	Ground
Y10	VSS	Ground
Y16	VSS	Ground
Y24	VSS	Ground
Y26	VSS	Ground
Y28	VSS	Ground
AA15	VSS	Ground
AA21	VSS	Ground
AA25	VSS	Ground
AA29	VSS	Ground
AA33	VSS	Ground
AA37	VSS	Ground
AB10	VSS	Ground
AB12	VSS	Ground
AB20	VSS	Ground
AB24	VSS	Ground
AB28	VSS	Ground
AC13	VSS	Ground
AC15	VSS	Ground
AC19	VSS	Ground
AC23	VSS	Ground
AC27	VSS	Ground
AC31	VSS	Ground
AC37	VSS	Ground
AC39	VSS	Ground
AD12	VSS	Ground
AD20	VSS	Ground

Ball No	Ball Name	Description
AD24	VSS	Ground
AD28	VSS	Ground
AD30	VSS	Ground
AD38	VSS	Ground
AD40	VSS	Ground
AD42	VSS	Ground
AE9	VSS	Ground
AE17	VSS	Ground
AE25	VSS	Ground
AE33	VSS	Ground
AE37	VSS	Ground
AF12	VSS	Ground
AF16	VSS	Ground
AF22	VSS	Ground
AF24	VSS	Ground
AF30	VSS	Ground
AF36	VSS	Ground
AG9	VSS	Ground
AG11	VSS	Ground
AG15	VSS	Ground
AG21	VSS	Ground
AG27	VSS	Ground
AG29	VSS	Ground
AG31	VSS	Ground
AG35	VSS	Ground
AG37	VSS	Ground
AH12	VSS	Ground
AH16	VSS	Ground
AH22	VSS	Ground
AH26	VSS	Ground
AH28	VSS	Ground

Ball No	Ball Name	Description
AH34	VSS	Ground
AH36	VSS	Ground
AJ9	VSS	Ground
AJ17	VSS	Ground
AJ21	VSS	Ground
AJ27	VSS	Ground
AJ31	VSS	Ground
AJ35	VSS	Ground
AJ37	VSS	Ground
AJ41	VSS	Ground
AJ43	VSS	Ground
AK12	VSS	Ground
AK16	VSS	Ground
AK22	VSS	Ground
AK28	VSS	Ground
AK32	VSS	Ground
AK34	VSS	Ground
AL11	VSS	Ground
AL17	VSS	Ground
AL21	VSS	Ground
AL25	VSS	Ground
AL31	VSS	Ground
AL37	VSS	Ground
AM16	VSS	Ground
AM20	VSS	Ground
AM28	VSS	Ground
AM36	VSS	Ground
AN11	VSS	Ground
AN13	VSS	Ground
AN19	VSS	Ground
AN21	VSS	Ground

Ball No	Ball Name	Description
AN25	VSS	Ground
AN37	VSS	Ground
AN39	VSS	Ground
AP32	VSS	Ground
AP36	VSS	Ground
AP40	VSS	Ground
AP42	VSS	Ground
AT30	VSS	Ground
AV6	VSS	Ground
AV30	VSS	Ground
AV34	VSS	Ground
AV36	VSS	Ground
AV40	VSS	Ground
AY34	VSS	Ground
BA37	VSS	Ground
AN23	VSS	Ground
NC		
A1	NC	Not Connect
A43	NC	Not Connect
BA1	NC	Not Connect
BA43	NC	Not Connect

3.5 Pin Multiplexed Function List

Table 3-3 Pin Multiplexed Function List

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
BA13	PDWUWS WCDGSDE	SIMCLK0	SIMCLK	O					GPIO1	I/O/T
BA11	PDWUWS WCDGSDE	SIMDATA0	SIMDATA	I/O/T					GPIO2	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
AY10	PDWUWS WCDGSDE	SIMRST0	SIMRST0	O					GPIO3	I/O/T
AP10	PDWUWS WCDGSDE	SIMCLK1	SIMCLK1	O					GPIO4	I/O/T
AT10	PDWUWS WCDGSDE	SIMDATA1	SIMDATA1	I/O/T					GPIO5	I/O/T
AR11	PDWUWS WCDGSDE	SIMRST1	SIMRST1	O					GPIO6	I/O/T
F26	SPSCBC2_8X_W	KEYIN0	KEYIN0	I/O/T	EXTINT2	I/O/T	SE_GPIO4	I/O/T	GPIO7	I/O/T
B26	SPSCBC2_8X_W	KEYIN1	KEYIN1	I/O/T	EXTINT3	I/O/T	SE_GPIO5	I/O/T	GPIO8	I/O/T
E25	SPSCBC2_8X_W	KEYIN2	KEYIN2	I/O/T	EXTINT4	I/O/T			GPIO9	I/O/T
AP22	SPSCBC2_8X	PWMA	PWMA	O					GPIO10	I/O/T
AV22	SPSCBC2_8X_W	LCM0_RSTN	LCM0_RSTN	O					GPIO11	I/O/T
AW21	SPSCBC2_8X_W	DSI0_TE	DSI0_TE	I					GPIO12	I/O/T
AV20	SPSCBC2_8X_W	EXTINT0	EXTINT0	I	KEYOUT5	I/O/T	SE_GPIO14	I/O/T	GPIO13	I/O/T
AU21	SPSCBC2_8X_W	EXTINT1	EXTINT1	I	KEYOUT6	I/O/T	SE_GPIO15	I/O/T	GPIO14	I/O/T
AR23	SPSCBC2_8X_W	SDA3	SDA3	I/O/T					GPIO15	I/O/T
AP24	SPSCBC2_8X_W	SCL3	SCL3	I/O/T					GPIO16	I/O/T
AV8	PDWUWS WCDGSDE	SD0_D3	SD0_D3	I/O/T					GPIO17	I/O/T
AU9	PDWUWS WCDGSDE	SD0_D2	SD0_D2	I/O/T					GPIO18	I/O/T
AW9	PDWUWS WCDGSDE	SD0_CMD	SD0_CMD	I/O/T					GPIO19	I/O/T
BA9	PDWUWS WCDGSDE	SD0_D0	SD0_D0	I/O/T					GPIO20	I/O/T
BA7	PDWUWS WCDGSDE	SD0_D1	SD0_D1	I/O/T					GPIO21	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
AY8	PDWUWS WCDGSDE	SD0_CLK	SD0_CLK	O/T					GPIO2 2	I/O/T
AT40	SPSEBC2_24X	EMMC_D0	EMMC_D0	I/O/T					GPIO2 3	I/O/T
AR41	SPSEBC2_24X	EMMC_CMD	EMMC_CMD	I/O/T					GPIO2 4	I/O/T
AY42	SPSEBC2_24X	EMMC_D6	EMMC_D6	I/O/T					GPIO2 5	I/O/T
AW41	SPSEBC2_24X	EMMC_D7	EMMC_D7	I/O/T					GPIO2 6	I/O/T
AU43	SPSEBC2_24X	EMMC_CLK	EMMC_CLK	O/T					GPIO2 7	I/O/T
AR39	SPSEBC2_24X	EMMC_D5	EMMC_D5	I/O/T					GPIO2 8	I/O/T
AV42	SPSEBC2_24X	EMMC_D4	EMMC_D4	I/O/T					GPIO2 9	I/O/T
AT42	SPSEBC2_24X	EMMC_DS	EMMC_DS	I					GPIO3 0	I/O/T
AP38	SPSEBC2_24X	EMMC_D3	EMMC_D3	I/O/T					GPIO3 1	I/O/T
AR43	SPSEBC2_24X	EMMC_RST	EMMC_RST	O					GPIO3 2	I/O/T
AW43	SPSEBC2_24X	EMMC_D1	EMMC_D1	I/O/T					GPIO3 3	I/O/T
AU41	SPSEBC2_24X	EMMC_D2	EMMC_D2	I/O/T					GPIO3 4	I/O/T
A17	SPSCBC2_8X	CMMC_LK0	CMMC_LK0	O			DBG_B US0(G1)	I/O/T	GPIO3 5	I/O/T
C17	SPSCBC2_8X	CMMC_LK1	CMMC_LK1	O			DBG_B US1(G1)	I/O/T	GPIO3 6	I/O/T
B18	SPSCBC2_8X	CMMC_LK2	CMMC_LK2	O	EXTINT 7(G1)	I/O/T	DBG_B US2(G1)	I/O/T	GPIO3 7	I/O/T
E17	SPSCBC2_8X	CMMC_LK3	CMMC_LK3	O	CLK_A UX3	O	DBG_B US3(G1)	I/O/T	GPIO3 8	I/O/T
J3	SPSCBC2_8X	CMRST0	CMRST0	I/O/T			DBG_B US4(G1)	I/O/T	GPIO3 9	I/O/T
K6	SPSCBC2_8X	CMRST1	CMRST1	I/O/T			DBG_B US5(G1)	I/O/T	GPIO4 0	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
J7	SPSCBC2_8X	CMRST2	CMRST2	I/O/T			DBG_BUS6(G1)	I/O/T	GPIO41	I/O/T
L1	SPSCBC2_8X	CMRST3	CMRST3	I/O/T			DBG_BUS7(G1)	I/O/T	GPIO42	I/O/T
J5	SPSCBC2_8X	CMRST4	CMRST4	I/O/T	KEYIN3	I/O/T	DBG_BUS8(G1)	I/O/T	GPIO43	I/O/T
K2	SPSCBC2_8X	CMRST5	CMRST5	I/O/T	KEYIN4	I/O/T	DBG_BUS9(G1)	I/O/T	GPIO44	I/O/T
G17	SPSCBC2_8X	CMPD0	CMPD0	I/O/T			DBG_BUS10(G1)	I/O/T	GPIO45	I/O/T
K4	SPSCBC2_8X	CMPD1	CMPD1	I/O/T			DBG_BUS11(G1)	I/O/T	GPIO46	I/O/T
D18	SPSCBC2_8X	CMPD2	CMPD2	I/O/T			DBG_BUS12(G1)	I/O/T	GPIO47	I/O/T
H16	SPSCBC2_8X	CMPD3	CMPD3	I/O/T			DBG_BUS13(G1)	I/O/T	GPIO48	I/O/T
F18	SPSCBC2_8X	CMPD4	CMPD4	I/O/T	KEYIN5	I/O/T	DBG_BUS14(G1)	I/O/T	GPIO49	I/O/T
H18	SPSCBC2_8X	CMPD5	CMPD5	I/O/T	KEYIN6	I/O/T	DBG_BUS15(G1)	I/O/T	GPIO50	I/O/T
A19	SPSCBC2_8X	SCL0	SCL0	I/O/T			DBG_BUS16(G1)	I/O/T	GPIO51	I/O/T
B20	SPSCBC2_8X	SDA0	SDA0	I/O/T			DBG_BUS17(G1)	I/O/T	GPIO52	I/O/T
C19	SPSCBC2_8X	SCL1	SCL1	I/O/T			DBG_BUS18(G1)	I/O/T	GPIO53	I/O/T
E19	SPSCBC2_8X	SDA1	SDA1	I/O/T			DBG_BUS19(G1)	I/O/T	GPIO54	I/O/T
C21	SPSCBC2_8X	SCL8	SCL8	I/O/T	KEYOUT3	I/O/T	DBG_BUS20(G1)	I/O/T	GPIO55	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
D20	SPSCBC2_8X	SDA8	SDA8	I/O/T	KEYOUT4	I/O/T	DBG_BUS21(G1)	I/O/T	GPIO56	I/O/T
H20	SPSCBC2_8X	SCL9	SCL9	I/O/T	EXTINT14(G0)	I/O/T	DBG_BUS22(G1)	I/O/T	GPIO57	I/O/T
G19	SPSCBC2_8X	SDA9	SDA9	I/O/T	EXTINT15(G0)	I/O/T	DBG_BUS23(G1)	I/O/T	GPIO58	I/O/T
A21	SPSCBC2_8X	SPI3_CS_N	SPI3_CS_N	I/O/T			ESE_GPIO6	I/O/T	GPIO59	I/O/T
B22	SPSCBC2_8X	SPI3_CLK	SPI3_CLK	I/O/T			ESE_GPIO7	I/O/T	GPIO60	I/O/T
E21	SPSCBC2_8X	SPI3_DI	SPI3_DI	I/O/T			CM4_GPIO2	I/O/T	GPIO61	I/O/T
D22	SPSCBC2_8X	SPI3_DO	SPI3_DO	I/O/T			ESE_GPIO5	I/O/T	GPIO62	I/O/T
AJ5	SPSCBC2_8X_SR	RFSPISDA0	RFSPISDA0	I/O/T					GPIO63	I/O/T
AH4	SPSCBC2_8X_SR	RFSPISDA1	RFSPISDA1	I/O/T					GPIO64	I/O/T
AJ3	SPSCBC2_8X_SR	RFSPISDA2	RFSPISDA2	I/O/T					GPIO65	I/O/T
AK2	SPSCBC2_8X_SR	RFSPISDA3	RFSPISDA3	I/O/T					GPIO66	I/O/T
AJ1	SPSCBC2_8X_SR	RFSPISCK	RFSPISCK	O/T					GPIO67	I/O/T
AH2	SPSCBC2_8X_SR	RFSPISSEN	RFSPISSEN	I/O/T					GPIO68	I/O/T
AP6	SPSCBC2_8X	RFCTL0(G0)	RFCTL0(G0)	I/O/T			DBG_BUS0(G0)	I/O/T	GPIO69	I/O/T
AP8	SPSCBC2_8X	RFCTL1(G0)	RFCTL1(G0)	I/O/T			DBG_BUS1(G0)	I/O/T	GPIO70	I/O/T
AR7	SPSCBC2_8X	RFCTL2(G0)	RFCTL2(G0)	I/O/T			DBG_BUS2(G0)	I/O/T	GPIO71	I/O/T
AU7	SPSCBC2_8X	RFCTL3(G0)	RFCTL3(G0)	I/O/T			DBG_BUS3(G0)	I/O/T	GPIO72	I/O/T
AR5	SPSCBC2_8X	RFCTL4(G0)	RFCTL4(G0)	I/O/T			DBG_BUS4(G0)	I/O/T	GPIO73	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
AT6	SPSCBC2_8X	RFCTL5 (G0)	RFCTL5 (G0)	I/O/T			DBG_B US5(G0)	I/O/T	GPIO7 4	I/O/T
AU5	SPSCBC2_8X	RFCTL6 (G0)	RFCTL6 (G0)	I/O/T			DBG_B US6(G0)	I/O/T	GPIO7 5	I/O/T
AT4	SPSCBC2_8X	RFCTL7 (G0)	RFCTL7 (G0)	I/O/T			DBG_B US7(G0)	I/O/T	GPIO7 6	I/O/T
AU3	SPSCBC2_8X	RFCTL8 (G0)	RFCTL8 (G0)	I/O/T			DBG_B US8(G0)	I/O/T	GPIO7 7	I/O/T
AV4	SPSCBC2_8X	RFCTL9 (G0)	RFCTL9 (G0)	I/O/T			DBG_B US9(G0)	I/O/T	GPIO7 8	I/O/T
AY4	SPSCBC2_8X	RFCTL10 (G0)	RFCTL10 (G0)	I/O/T			DBG_B US10(G0)	I/O/T	GPIO7 9	I/O/T
AY2	SPSCBC2_8X	RFCTL11 (G0)	RFCTL11 (G0)	I/O/T			DBG_B US11(G0)	I/O/T	GPIO8 0	I/O/T
AW3	SPSCBC2_8X	RFCTL12 (G0)	RFCTL12 (G0)	I/O/T			DBG_B US12(G0)	I/O/T	GPIO8 1	I/O/T
AW1	SPSCBC2_8X	RFCTL13 (G0)	RFCTL13 (G0)	I/O/T	WIFI_COEXIST	I/O/T	DBG_B US13(G0)	I/O/T	GPIO8 2	I/O/T
AV2	SPSCBC2_8X	RFCTL14 (G0)	RFCTL14 (G0)	I/O/T	UART_matrix.INF3.TXD	O	DBG_B US14(G0)	I/O/T	GPIO8 3	I/O/T
AU1	SPSCBC2_8X	RFCTL15 (G0)	RFCTL15 (G0)	I/O/T	UART_matrix.INF3.RXD	I	DBG_B US15(G0)	I/O/T	GPIO8 4	I/O/T
AL5	SPSCBC2_8X_SR	RFFE0_SCK	RFFE0_SCK	I/O/T	RFCTL0 (G1)	I/O/T	DBG_B US16(G0)	I/O/T	GPIO8 5	I/O/T
AM4	SPSCBC2_8X_SR	RFFE0_SDA	RFFE0_SDA	I/O/T	RFCTL1 (G1)	I/O/T	DBG_B US17(G0)	I/O/T	GPIO8 6	I/O/T
AK6	SPSCBC2_8X_SR	RFFE1_SCK	RFFE1_SCK	I/O/T	RFCTL2 (G1)	I/O/T	DBG_B US18(G0)	I/O/T	GPIO8 7	I/O/T
AK8	SPSCBC2_8X_SR	RFFE1_SDA	RFFE1_SDA	I/O/T	RFCTL3 (G1)	I/O/T	DBG_B US19(G0)	I/O/T	GPIO8 8	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
AK4	SPSCBC2_8X_SR	RFFE2_SCK	RFFE2_SCK	I/O/T	RFCTL4(G1)	I/O/T	DBG_BUS20(G0)	I/O/T	GPIO89	I/O/T
AL3	SPSCBC2_8X_SR	RFFE2_SDA	RFFE2_SDA	I/O/T	RFCTL5(G1)	I/O/T	DBG_BUS21(G0)	I/O/T	GPIO90	I/O/T
AN5	SPSCBC2_8X_SR	RFFE3_SCK	RFFE3_SCK	I/O/T	RFCTL6(G1)	I/O/T	DBG_BUS22(G0)	I/O/T	GPIO91	I/O/T
AN3	SPSCBC2_8X_SR	RFFE3_SDA	RFFE3_SDA	I/O/T	RFCTL7(G1)	I/O/T	DBG_BUS23(G0)	I/O/T	GPIO92	I/O/T
AP4	SPSCBC2_8X_SR	RFFE4_SCK	RFFE4_SCK	I/O/T	RFCTL8(G1)	I/O/T	DBG_BUS24(G0)	I/O/T	GPIO93	I/O/T
AR3	SPSCBC2_8X_SR	RFFE4_SDA	RFFE4_SDA	I/O/T	RFCTL9(G1)	I/O/T	DBG_BUS25(G0)	I/O/T	GPIO94	I/O/T
AT2	SPSCBC2_8X_SR	RFFE5_SCK	RFFE5_SCK	I/O/T	RFCTL10(G1)	I/O/T	DBG_BUS26(G0)	I/O/T	GPIO95	I/O/T
AR1	SPSCBC2_8X_SR	RFFE5_SDA	RFFE5_SDA	I/O/T	RFCTL11(G1)	I/O/T	DBG_BUS27(G0)	I/O/T	GPIO96	I/O/T
AN1	SPSCBC2_8X_SR	RFFE6_SCK	RFFE6_SCK	I/O/T	RFCTL12(G1)	I/O/T	DBG_BUS28(G0)	I/O/T	GPIO97	I/O/T
AP2	SPSCBC2_8X_SR	RFFE6_SDA	RFFE6_SDA	I/O/T	RFCTL13(G1)	I/O/T	DBG_BUS29(G0)	I/O/T	GPIO98	I/O/T
AM2	SPSCBC2_8X_SR	RFFE7_SCK	RFFE7_SCK	I/O/T	RFCTL14(G1)	I/O/T	DBG_BUS30(G0)	I/O/T	GPIO99	I/O/T
AL1	SPSCBC2_8X_SR	RFFE7_SDA	RFFE7_SDA	I/O/T	RFCTL15(G1)	I/O/T	DBG_BUS31(G0)	I/O/T	GPIO100	I/O/T
AH6	SPSCBC2_8X	LVDSRF_ADDAC0_ON	LVDSRF_ADDAC0_ON	I/O/T					GPIO101	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
AJ7	SPSCBC2_8X	LVDSR_F_ADDAC1_ON	LVDSR_F_ADDAC1_ON	I/O/T					GPIO102	I/O/T
AH8	SPSCBC2_8X	LVDSR_F_ADDAC2_ON	LVDSR_F_ADDAC2_ON	I/O/T					GPIO103	I/O/T
AG7	SPSCBC2_8X	LVDSR_F_ADDAC3_ON	LVDSR_F_ADDAC3_ON	I/O/T					GPIO104	I/O/T
A23	SPSCBC2_8X	SPI0_CS_N	SPI0_CS_N	I/O/T			EXTINT5(G0)	I/O/T	GPIO105	I/O/T
B24	SPSCBC2_8X	SPI0_DO	SPI0_DO	I/O/T			EXTINT6(G0)	I/O/T	GPIO106	I/O/T
C23	SPSCBC2_8X	SPI0_DI	SPI0_DI	I/O/T			EXTINT7(G0)	I/O/T	GPIO107	I/O/T
A25	SPSCBC2_8X	SPI0_CLK	SPI0_CLK	I/O/T			EXTINT8(G0)	I/O/T	GPIO108	I/O/T
AV12	SPSCBC2_8X	IIS1DI	IIS1DI	I	SE_GPIO0	I/O/T	EXTINT13	I/O/T	GPIO109	I/O/T
AW11	SPSCBC2_8X	IIS1DO	IIS1DO	O/T	SE_GPIO1	I/O/T	CM4_GPIO3(G0)	I/O/T	GPIO110	I/O/T
AV10	SPSCBC2_8X	IIS1CLK	IIS1CLK	I/O/T	SE_GPIO2	I/O/T	CM4_GPIO4(G0)	I/O/T	GPIO111	I/O/T
AU11	SPSCBC2_8X	IIS1LRC_K	IIS1LRC_K	I/O/T	SE_GPIO3	I/O/T			GPIO112	I/O/T
AT12	SPSCBC2_8X	SCL4	SCL4	I/O/T					GPIO113	I/O/T
AP12	SPSCBC2_8X	SDA4	SDA4	I/O/T					GPIO114	I/O/T
AY14	SPSCBC2_8X	CLK_AUX1	CLK_AUX1	O					GPIO115	I/O/T
BA5	PDWUWS_WCDGSDE	SD2_CLK	SD2_CLK	O/T	SPI1_CLK	I/O/T	SE_GPIO11	I/O/T	GPIO116	I/O/T
AY6	PDWUWS_WCDGSDE	SD2_CMD	SD2_CMD	I/O/T	SPI1_DI	I/O/T	SE_GPIO12	I/O/T	GPIO117	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
AW5	PDWUWS WCDGSDE	SD2_D0	SD2_D0	I/O/T	SPI1_D O	I/O/T	SE_GPI O13	I/O/T	GPIO1 18	I/O/T
AW7	PDWUWS WCDGSDE	SD2_D1	SD2_D1	I/O/T	SE_GPI O7	I/O/T			GPIO1 19	I/O/T
AR9	PDWUWS WCDGSDE	SD2_D2	SD2_D2	I/O/T	SE_GPI O8	I/O/T	EXTINT 8(G1)	I/O/T	GPIO1 20	I/O/T
AT8	PDWUWS WCDGSDE	SD2_D3	SD2_D3	I/O/T	SPI1_CS N	I/O/T	SE_GPI O9	I/O/T	GPIO1 21	I/O/T
F20	SPSCBC2_ 8X	SCL2	SCL2	I/O/T					GPIO1 22	I/O/T
G21	SPSCBC2_ 8X	SDA2	SDA2	I/O/T					GPIO1 23	I/O/T
B30	SPSCBC2_ 8X_W	KEYOU T1	KEYOU T1	I/O/T	EXTINT 12	I/O/T	CM4_G PIO6	I/O/T	GPIO1 24	I/O/T
A31	SPSCBC2_ 8X_W	KEYOU T0	KEYOU T0	I/O/T	EXTINT 11	I/O/T	CM4_G PIO5	I/O/T	GPIO1 25	I/O/T
A29	SPSCBC2_ 8X_W	KEYOU T2	KEYOU T2	I/O/T	PWMB	I/O/T	CM4_G PIO7	I/O/T	GPIO1 26	I/O/T
A27	SPSCBC2_ 8X	EXTINT 9	EXTINT 9	I/O/T	CM4_G PIO0	I/O/T			GPIO1 27	I/O/T
B28	SPSCBC2_ 8X	EXTINT 10	EXTINT 10	I/O/T	CM4_G PIO1	I/O/T			GPIO1 28	I/O/T
W7	SPSCBC2_ 8X	SCL6	SCL6	I/O/T					GPIO1 29	I/O/T
V8	SPSCBC2_ 8X	SDA6	SDA6	I/O/T					GPIO1 30	I/O/T
V4	SPSCBC2_ 8X	VDSP_T DO	VDSP_T DO	O/T			ESE_GP IO0	I/O/T	GPIO1 31	I/O/T
V6	SPSCBC2_ 8X	VDSP_T DI	VDSP_T DI	I/O/T			ESE_GP IO1	I/O/T	GPIO1 32	I/O/T
Y6	SPSCBC2_ 8X	VDSP_T MS	VDSP_T MS	I/O/T			ESE_GP IO2	I/O/T	GPIO1 33	I/O/T
U3	SPSCBC2_ 8X	VDSP_T CK	VDSP_T CK	I			ESE_GP IO3	I/O/T	GPIO1 34	I/O/T
V2	SPSCBC2_ 8X	VDSP_ RTCK	VDSP_ RTCK	I/O/T			ESE_GP IO4	I/O/T	GPIO1 35	I/O/T
AY1 2	SPSCBC2_ 8X_W	SIM_DE T0	SIM_DE T0	I/O/T					GPIO1 36	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
AW13	SPSCBC2_8X_W	SIM_DET1	SIM_DET1	I/O/T					GPIO137	I/O/T
AU13	SPSCBC2_8X_W	TF_DET	TF_DET	I/O/T					GPIO138	I/O/T
AV14	SPSCBC2_8X	BAT_DET	BAT_DET	I/O/T					GPIO139	I/O/T
U1	SPSCBC2_8X	U2TXD	U2TXD	O			DBG_BUS26(G1)	I/O/T	GPIO140	I/O/T
T2	SPSCBC2_8X	U2RXD	U2RXD	I			DBG_BUS27(G1)	I/O/T	GPIO141	I/O/T
T4	SPSCBC2_8X	U1TXD	U1TXD	O			DBG_BUS24(G1)	I/O/T	GPIO142	I/O/T
R3	SPSCBC2_8X	U1RXD	U1RXD	I	WDRST	I/O/T	DBG_BUS25(G1)	I/O/T	GPIO143	I/O/T
M6	SPSCBC2_8X	MTCK_ARM	MTCK_ARM	I					GPIO144	I/O/T
L5	SPSCBC2_8X	MTMS_ARM	MTMS_ARM	I/O/T					GPIO145	I/O/T
P4	SPSCBC2_8X	AUD_DSP_TDO	AUD_DSP_TDO	I/O/T			C2K_DSP_TDO	I/O/T	GPIO146	I/O/T
P2	SPSCBC2_8X	AUD_DSP_RTC_K	AUD_DSP_RTC_K	O	SE_GPIO6	I/O/T	C2K_DSP_RTC_K	I/O/T	GPIO147	I/O/T
N1	SPSCBC2_8X	AUD_DSP_TDI	AUD_DSP_TDI	I/O/T			C2K_DSP_TDI	I/O/T	GPIO148	I/O/T
M2	SPSCBC2_8X	AUD_DSP_TCK	AUD_DSP_TCK	I	PWMD	I/O/T	C2K_DSP_TCK	I	GPIO149	I/O/T
R1	SPSCBC2_8X	AUD_DSP_TMS	AUD_DSP_TMS	I/O/T	SE_GPIO10	I/O/T	C2K_DSP_TMS	I/O/T	GPIO150	I/O/T
C27	SPSEBC2_24X	SD1_CMD	SD1_CMD	I/O/T					GPIO151	I/O/T
C29	SPSEBC2_24X	SD1_D0	SD1_D0	I/O/T	LCM1_RSTN	I/O/T			GPIO152	I/O/T
D28	SPSEBC2_24X	SD1_D1	SD1_D1	I/O/T	DSI1_TE	I/O/T			GPIO153	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
D26	SPSEBC2_24X	SD1_CLK	SD1_CLK	O/T	PCIE3_RST	I/O/T			GPIO154	I/O/T
C25	SPSEBC2_24X	SD1_D2	SD1_D2	I/O/T	PCIE3_CLKREQ	I/O/T			GPIO155	I/O/T
E27	SPSEBC2_24X	SD1_D3	SD1_D3	I/O/T	PCIE3_WAKE	I/O/T			GPIO156	I/O/T
B38	SPSCBC2_8X	CLK_AUX0	CLK_AUX0	O	PROBE_CLK	I/O/T			GPIO157	I/O/T
A37	SPSCBC2_8X	U0TXD	U0TXD	O					GPIO158	I/O/T
A35	SPSCBC2_8X	U0RXD	U0RXD	I					GPIO159	I/O/T
B36	SPSCBC2_8X	U0CTS	U0CTS	I					GPIO160	I/O/T
B34	SPSCBC2_8X	U0RTS	U0RTS	O					GPIO161	I/O/T
A41	SPSCBC2_8X	U4TXD	U4TXD	O					GPIO162	I/O/T
B42	SPSCBC2_8X	U4RXD	U4RXD	I					GPIO163	I/O/T
C41	SPSCBC2_8X	U4CTS	U4CTS	I					GPIO164	I/O/T
C39	SPSCBC2_8X	U4RTS	U4RTS	O					GPIO165	I/O/T
B32	SPSCBC2_8X	IIS0DI	IIS0DI	I					GPIO166	I/O/T
C31	SPSCBC2_8X	IIS0DO	IIS0DO	O/T					GPIO167	I/O/T
A33	SPSCBC2_8X	IIS0CLK	IIS0CLK	I/O/T					GPIO168	I/O/T
E31	SPSCBC2_8X	IIS0LRC	IIS0LRC	I/O/T					GPIO169	I/O/T
G27	SPSCBC2_8X	PWMC	PWMC	I/O/T	CM4_GPIO3(G1)	I/O/T			GPIO170	I/O/T
G29	SPSCBC2_8X	IIS4DO	IIS4DO	O/T					GPIO171	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
D30	SPSCBC2_8X	IIS4DI	IIS4DI	I					GPIO172	I/O/T
F28	SPSCBC2_8X	IIS4CLK	IIS4CLK	I/O/T					GPIO173	I/O/T
E29	SPSCBC2_8X	IIS4LRC	IIS4LRC	I/O/T	CM4_GPIO4(G1)	I/O/T			GPIO174	I/O/T
B40	SPSCBC2_8X	U5TXD	U5TXD	O	SCL5	I/O/T			GPIO175	I/O/T
A39	SPSCBC2_8X	U5RXD	U5RXD	I	SDA5	I/O/T			GPIO176	I/O/T
F24	SPSCBC2_8X	SPI2_CS	SPI2_CS	I/O/T	EXTINT5(G1)	I/O/T			GPIO177	I/O/T
F22	SPSCBC2_8X	SPI2_DO	SPI2_DO	I/O/T					GPIO178	I/O/T
E23	SPSCBC2_8X	SPI2_DI	SPI2_DI	I/O/T					GPIO179	I/O/T
G23	SPSCBC2_8X	SPI2_CLK	SPI2_CLK	I/O/T	EXTINT6(G1)	I/O/T			GPIO180	I/O/T
M4	SPSCBC2_8X	U7TXD	U7TXD	O			DBG_BUS30(G1)	I/O/T	GPIO181	I/O/T
L3	SPSCBC2_8X	U7RXD	U7RXD	I			DBG_BUS31(G1)	I/O/T	GPIO182	I/O/T
N3	SPSCBC2_8X	U6TXD	U6TXD	O	PLL_LOCK	I/O/T	DBG_BUS28(G1)	I/O/T	GPIO183	I/O/T
N5	SPSCBC2_8X	U6RXD	U6RXD	I	MGPLL_SWITCH	I/O/T	DBG_BUS29(G1)	I/O/T	GPIO184	I/O/T
R7	SPSCBC2_8X_W	DMIC_CLK0	DMIC_CLK0	O			IIS5DO	O/T	GPIO185	I/O/T
P6	SPSCBC2_8X_W	DMIC_DAT0	DMIC_DAT0	I/O/T			IIS5DI	I	GPIO186	I/O/T
T6	SPSCBC2_8X_W	DMIC_CLK1	DMIC_CLK1	O			IIS5CLK	I/O/T	GPIO187	I/O/T
U7	SPSCBC2_8X_W	DMIC_DAT1	DMIC_DAT1	I/O/T			IIS5LRC	I/O/T	GPIO188	I/O/T

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
U5	SPSCBC2_8X_W	DMIC_CLK2	DMIC_CLK2	O					GPIO189	I/O/T
R5	SPSCBC2_8X_W	DMIC_DAT2	DMIC_DAT2	I/O/T					GPIO190	I/O/T
AM6	SPSCBC2_8X	IIS3DI	IIS3DI	I	EXTINT14(G1)	I/O/T			GPIO191	I/O/T
AL7	SPSCBC2_8X	IIS3DO	IIS3DO	O/T	EXTINT15(G1)	I/O/T			GPIO192	I/O/T
AM8	SPSCBC2_8X	IIS3LRCK	IIS3LRCK	I/O/T	SDA7	I/O/T			GPIO193	I/O/T
AN7	SPSCBC2_8X	IIS3CLK	IIS3CLK	I/O/T	SCL7	I/O/T			GPIO194	I/O/T
BA3	SPSCBC2_8X	CLK_AUX2	CLK_AUX2	O					GPIO195	I/O/T
AU15	SPSCBC2_8X	CHG_TYPE	CHG_TYPE	I					GPIO196	I/O/T
AT18	SPSCBC2_8X	DNS_D0	DNS_D0	O	DNS_PGA_LVLL	O			GPIO197	I/O/T
AV18	SPSCBC2_8X	DNS_D1	DNS_D1	O	DNS_PGA_LVLR	O			GPIO198	I/O/T
BA21	SPSCBC2_8X	ADI_D	ADI_D	I/O						
AY22	SPSCBC2_8X	ADI_SCLK	ADI_SCLK	O						
AW17	SPSCBC2_8X	ANA_INT0	ANA_INT0	I						
AW19	SPSCBC2_8X	ANA_INT1	ANA_INT1	I						
AR21	SPSCBC2_8X	ANA_INT2	ANA_INT2	I						
AY16	SPSCBC2_8X	AUD_ADD0	AUD_ADD0	I	IIS2DI	I				
AU17	SPSCBC2_8X	AUD_ADD1	AUD_ADD1	I	AON_ADCSW	O				
AR17	SPSCBC2_8X	AUD_ADSYNC	AUD_ADSYNC	I						

Ball No	Cell Name	Ball name	Function0	Type	Function1	Type	Function2	Type	Function3	Type
AV16	SPSCBC2_8X	AUD_DAD0	AUD_DAD0	O	IIS2DO	O/T				
AY18	SPSCBC2_8X	AUD_DAD1	AUD_DAD1	O						
BA17	SPSCBC2_8X	AUD_DASYNC	AUD_DASYNC	O	IIS2LRC	I/O/T				
AT16	SPSCBC2_8X	AUD_SCLK	AUD_SCLK	O	IIS2CLK	O/T				
AT20	SPSCBC2_8X	CHIP_SLEEP	CHIP_SLEEP	O						
AY20	SPSCBC2_8X	CLK_32K	CLK_32K	I						
BA19	SPSCBC2_8X	DCDCCPU0_EN	DCDCCPU0_EN	O						
AU19	SPSCBC2_8X	DCDCCPU1_EN	DCDCCPU1_EN	O						
AP16	SPSCBC2_8X	DCDCCPU2_EN	DCDCCPU2_EN	O						
AP20	SPSCBC2_8X	EXT_RST_B	EXT_RST_B	I						
BA15	SPSCBC2_8X	XTL_EN0	XTL_EN0	O						
AW15	SPSCBC2_8X	XTL_EN1	XTL_EN1	O						
AR19	SPSCBC2_8X	XTL_EN2	XTL_EN2	O						
AT22	SPSCBC2_8X_W	SSUSB_SW	SSUSB_SW	I						
D38	SPSCBC2_8X	ESE_GPIO	ESE_GPIO	I/O/T						

3.6 Pin Status and GPIOs

Ball No	Ball Name	GPIO	At Reset		After Reset		Power Domain
			H/L/Hiz	Pin State	H/L/Hiz/X	Pin State	
BA13	SIMCLK0	GPIO1	L	INPUT(WPD)	L	INPUT(WPD)	VSIM0
BA11	SIMDAT0	GPIO2	L	INPUT(WPD)	L	INPUT(WPD)	VSIM0
AY10	SIMRST0	GPIO3	L	INPUT(WPD)	L	INPUT(WPD)	VSIM0
AP10	SIMCLK1	GPIO4	L	INPUT(WPD)	L	INPUT(WPD)	VSIM1
AT10	SIMDAT1	GPIO5	L	INPUT(WPD)	L	INPUT(WPD)	VSIM1
AR11	SIMRST1	GPIO6	L	INPUT(WPD)	L	INPUT(WPD)	VSIM1
E25	KEYIN2	GPIO9	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AP22	PWMA	GPIO10	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AV22	LCM0_RSTN	GPIO11	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AW21	DSI0_TE	GPIO12	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AV20	EXTINT0	GPIO13	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AU21	EXTINT1	GPIO14	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AR23	SDA3	GPIO15	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
AP24	SCL3	GPIO16	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
AV8	SD0_D3	GPIO17	L	INPUT(WPD)	X	INPUT(WPD)	VSD0
AU9	SD0_D2	GPIO18	L	INPUT(WPD)	X	INPUT(WPD)	VSD0
AW9	SD0_CMD	GPIO19	L	INPUT(WPD)	X	INPUT(WPD)	VSD0
BA9	SD0_D0	GPIO20	L	INPUT(WPD)	X	INPUT(WPD)	VSD0
BA7	SD0_D1	GPIO21	L	INPUT(WPD)	X	INPUT(WPD)	VSD0
AY8	SD0_CLK	GPIO22	L	INPUT(WPD)	X	INPUT(WPD)	VSD0
AT40	EMMC_D0	GPIO23	H	INPUT(WPU)	H	INPUT(WPU)	VEMMC IO
AR41	EMMC_CMD	GPIO24	H	INPUT(WPU)	H	INPUT(WPU)	VEMMC IO
AY42	EMMC_D6	GPIO25	H	INPUT(WPU)	H	INPUT(WPU)	VEMMC IO
AW41	EMMC_D7	GPIO26	H	INPUT(WPU)	H	INPUT(WPU)	VEMMC IO
AU43	EMMC_CLK	GPIO27	L	OUTPUT	L	OUTPUT	VEMMC IO

Ball No	Ball Name	GPIO	At Reset		After Reset		Power Domain
			H/L/Hiz	Pin State	H/L/Hiz/X	Pin State	
AR39	EMMC_D5	GPIO28	H	INPUT(WPU)	H	INPUT(WPU)	VEMMC IO
AV42	EMMC_D4	GPIO29	H	INPUT(WPU)	H	INPUT(WPU)	VEMMC IO
AT42	EMMC_DS	GPIO30	L	INPUT(WPD)	L	INPUT(WPD)	VEMMC IO
AP38	EMMC_D3	GPIO31	H	INPUT(WPU)	H	INPUT(WPU)	VEMMC IO
AR43	EMMC_RST	GPIO32	H	OUTPUT	H	OUTPUT	VEMMC IO
AW43	EMMC_D1	GPIO33	H	INPUT(WPU)	H	INPUT(WPU)	VEMMC IO
AU41	EMMC_D2	GPIO34	H	INPUT(WPU)	H	INPUT(WPU)	VEMMC IO
A17	CMMCLK0	GPIO35	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
C17	CMMCLK1	GPIO36	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
B18	CMMCLK2	GPIO37	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
E17	CMMCLK3	GPIO38	L	INPUT(WPD)	X	INPUT(WPD)	VIO1V8
J3	CMRST0	GPIO39	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
K6	CMRST1	GPIO40	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
J7	CMRST2	GPIO41	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
L1	CMRST3	GPIO42	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
J5	CMRST4	GPIO43	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
K2	CMRST5	GPIO44	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
G17	CMPD0	GPIO45	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
K4	CMPD1	GPIO46	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
D18	CMPD2	GPIO47	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
H16	CMPD3	GPIO48	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
F18	CMPD4	GPIO49	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
H18	CMPD5	GPIO50	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
A19	SCL0	GPIO51	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
B20	SDA0	GPIO52	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8

Ball No	Ball Name	GPIO	At Reset		After Reset		Power Domain
			H/L/Hiz	Pin State	H/L/Hiz/X	Pin State	
C19	SCL1	GPIO53	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
E19	SDA1	GPIO54	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
C21	SCL8	GPIO55	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
D20	SDA8	GPIO56	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
H20	SCL9	GPIO57	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
G19	SDA9	GPIO58	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
A21	SPI3_CSN	GPIO59	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
B22	SPI3_CLK	GPIO60	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
E21	SPI3_DI	GPIO61	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
D22	SPI3_DO	GPIO62	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AJ5	RFSPi_SDA0	GPIO63	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AH4	RFSPi_SDA1	GPIO64	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AJ3	RFSPi_SDA2	GPIO65	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AK2	RFSPi_SDA3	GPIO66	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AJ1	RFSPi_SCK	GPIO67	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AH2	RFSPi_SEN	GPIO68	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AP6	RFCTL0(G0)	GPIO69	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AP8	RFCTL1(G0)	GPIO70	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AR7	RFCTL2(G0)	GPIO71	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AU7	RFCTL3(G0)	GPIO72	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AR5	RFCTL4(G0)	GPIO73	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AT6	RFCTL5(G0)	GPIO74	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AU5	RFCTL6(G0)	GPIO75	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AT4	RFCTL7(G0)	GPIO76	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AU3	RFCTL8(G0)	GPIO77	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AV4	RFCTL9(G0)	GPIO78	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AY4	RFCTL10(G0)	GPIO79	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AY2	RFCTL11(G0)	GPIO80	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AW3	RFCTL12(G0)	GPIO81	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AW1	RFCTL13(G0)	GPIO82	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8

Ball No	Ball Name	GPIO	At Reset		After Reset		Power Domain
			H/L/Hiz	Pin State	H/L/Hiz/X	Pin State	
AV2	RFCTL14(G0)	GPIO83	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AU1	RFCTL15(G0)	GPIO84	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AL5	RFFE0_SCK	GPIO85	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AM4	RFFE0_SDA	GPIO86	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AK6	RFFE1_SCK	GPIO87	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AK8	RFFE1_SDA	GPIO88	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AK4	RFFE2_SCK	GPIO89	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AL3	RFFE2_SDA	GPIO90	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AN5	RFFE3_SCK	GPIO91	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AN3	RFFE3_SDA	GPIO92	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AP4	RFFE4_SCK	GPIO93	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AR3	RFFE4_SDA	GPIO94	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AT2	RFFE5_SCK	GPIO95	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AR1	RFFE5_SDA	GPIO96	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AN1	RFFE6_SCK	GPIO97	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AP2	RFFE6_SDA	GPIO98	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AM2	RFFE7_SCK	GPIO99	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AL1	RFFE7_SDA	GPIO100	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AH6	LVDSRF_AD DAC0_ON	GPIO101	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AJ7	LVDSRF_AD DAC1_ON	GPIO102	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AH8	LVDSRF_AD DAC2_ON	GPIO103	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AG7	LVDSRF_AD DAC3_ON	GPIO104	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
A23	SPI0_CSN	GPIO105	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
B24	SPI0_DO	GPIO106	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
C23	SPI0_DI	GPIO107	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
A25	SPI0_CLK	GPIO108	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AV12	IIS1DI	GPIO109	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8

Ball No	Ball Name	GPIO	At Reset		After Reset		Power Domain
			H/L/Hiz	Pin State	H/L/Hiz/X	Pin State	
AW11	IIS1DO	GPIO110	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AV10	IIS1CLK	GPIO111	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AU11	IIS1LRCK	GPIO112	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AT12	SCL4	GPIO113	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
AP12	SDA4	GPIO114	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
AY14	CLK_AUX1	GPIO115	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
BA5	SD2_CLK	GPIO116	L	INPUT(WPD)	L	INPUT(WPD)	VSD2
AY6	SD2_CMD	GPIO117	L	INPUT(WPD)	L	INPUT(WPD)	VSD2
AW5	SD2_D0	GPIO118	L	INPUT(WPD)	L	INPUT(WPD)	VSD2
AW7	SD2_D1	GPIO119	L	INPUT(WPD)	L	INPUT(WPD)	VSD2
AR9	SD2_D2	GPIO120	L	INPUT(WPD)	L	INPUT(WPD)	VSD2
AT8	SD2_D3	GPIO121	L	INPUT(WPD)	L	INPUT(WPD)	VSD2
F20	SCL2	GPIO122	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
G21	SDA2	GPIO123	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
B30	KEYOUT1	GPIO124	L	WPD	L	WPD	VIO1V8
A31	KEYOUT0	GPIO125	L	OUTPUT	L	OUTPUT	VIO1V8
A29	KEYOUT2	GPIO126	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
A27	EXTINT9	GPIO127	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
B28	EXTINT10	GPIO128	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
W7	SCL6	GPIO129	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
V8	SDA6	GPIO130	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
V4	VDSP_TDO	GPIO131	Hiz	Hiz	Hiz	Hiz	VIO1V8
V6	VDSP_TDI	GPIO132	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
Y6	VDSP_TMS	GPIO133	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
U3	VDSP_TCK	GPIO134	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
V2	VDSP_RTCK	GPIO135	L	OUTPUT	L	OUTPUT	VIO1V8
AY12	SIM_DET0	GPIO136	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AW13	SIM_DET1	GPIO137	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AU13	TF_DET	GPIO138	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AV14	BAT_DET	GPIO139	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8

Ball No	Ball Name	GPIO	At Reset		After Reset		Power Domain
			H/L/Hiz	Pin State	H/L/Hiz/X	Pin State	
U1	U2TXD	GPIO140	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
T2	U2RXD	GPIO141	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
T4	U1TXD	GPIO142	H	INPUT(WPU)	X	OUTPUT	VIO1V8
R3	U1RXD	GPIO143	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
M6	MTCK_ARM	GPIO144	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
L5	MTMS_ARM	GPIO145	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
P4	AUD_DSP_TD O	GPIO146	Hiz	Hiz	Hiz	Hiz	VIO1V8
P2	AUD_DSP_RT CK	GPIO147	L	OUTPUT	L	OUTPUT	VIO1V8
N1	AUD_DSP_TD I	GPIO148	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
M2	AUD_DSP_TC K	GPIO149	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
R1	AUD_DSP_T MS	GPIO150	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
C27	SD1_CMD	GPIO151	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
C29	SD1_D0	GPIO152	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
D28	SD1_D1	GPIO153	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
D26	SD1_CLK	GPIO154	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
C25	SD1_D2	GPIO155	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
E27	SD1_D3	GPIO156	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
B38	CLK_AUX0	GPIO157	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
A37	U0TXD	GPIO158	H	OUTPUT	X	OUTPUT	VIO1V8
A35	U0RXD	GPIO159	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
B36	U0CTS	GPIO160	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
B34	U0RTS	GPIO161	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
A41	U4TXD	GPIO162	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
B42	U4RXD	GPIO163	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
C41	U4CTS	GPIO164	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
C39	U4RTS	GPIO165	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
B32	IIS0DI	GPIO166	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8

Ball No	Ball Name	GPIO	At Reset		After Reset		Power Domain
			H/L/Hiz	Pin State	H/L/Hiz/X	Pin State	
C31	IIS0DO	GPIO167	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
A33	IIS0CLK	GPIO168	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
E31	IIS0LRCK	GPIO169	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
G27	PWMC	GPIO170	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
G29	IIS4DO	GPIO171	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
D30	IIS4DI	GPIO172	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
F28	IIS4CLK	GPIO173	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
E29	IIS4LRCK	GPIO174	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
B40	U5TXD	GPIO175	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
A39	U5RXD	GPIO176	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
F24	SPI2_CSN	GPIO177	H	INPUT(WPU)	H	INPUT(WPU)	VIO1V8
F22	SPI2_DO	GPIO178	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
E23	SPI2_DI	GPIO179	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
G23	SPI2_CLK	GPIO180	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
M4	U7TXD	GPIO181	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
L3	U7RXD	GPIO182	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
N3	U6TXD	GPIO183	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
N5	U6RXD	GPIO184	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
R7	DMIC_CLK0	GPIO185	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
P6	DMIC_DAT0	GPIO186	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
T6	DMIC_CLK1	GPIO187	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
U7	DMIC_DAT1	GPIO188	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
U5	DMIC_CLK2	GPIO189	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
R5	DMIC_DAT2	GPIO190	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AM6	IIS3DI	GPIO191	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AL7	IIS3DO	GPIO192	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AM8	IIS3LRCK	GPIO193	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AN7	IIS3CLK	GPIO194	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
BA3	CLK_AUX2	GPIO195	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AU15	CHG_TYPE	GPIO196	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8

Ball No	Ball Name	GPIO	At Reset		After Reset		Power Domain
			H/L/Hiz	Pin State	H/L/Hiz/X	Pin State	
AT18	DNS_D0	GPIO197	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8
AV18	DNS_D1	GPIO198	L	INPUT(WPD)	L	INPUT(WPD)	VIO1V8

NOTE

“X” means that the GPIO logic could be changed in the “After Reset”.

3.7 Pin Register Descriptions

See *UIS7885 Pin List and Pin Register Instructions*.

3.8 Strapping Pins

Ball name	Signal name in boot code	Value	Descriptions	Default value
U1TXD	USB_DLOAD_EN	0	USB download(no time out)	1(ie=1,wpu)
		1	UART download	
ADI_SCLK	ARM_BOOT_MD1	00	Reserved	11(ie=1,wpu) (MSB=MD1, LSB=MD0)
		01	UFS boot	
ADI_D	ARM_BOOT_MD0	10	SD boot(no time out)	
		11	eMMC 6.5MHz	
AUD_DAD0	ARM_BOOT_MD3	0	NON-KEY Download(With timeout) mode	1(ie=1,wpu)
		1	Normal mode	

Press key	Descriptions
Only Keyin0	USB download with timeout
Keyin0 + Keyin1	SD boot with timeout

3.9 Pad Information

The digital IO have 6 types:

- SPSCBC2_8X_VL/HL
- SPSCBC2_8X_W_VL/HL
- SPSCBC2_8X_SR_VL/HL
- SPSEBC2_24X_HL/VL
- PDWUWSWCDGSDE_V/H
- SPSEBC2_SWP_HL/VL

3.9.1 SPSCBC2_8X_VL/HL

SPSCBC2_8X_VL/HL is the normal pad used usually.

Truth Table of SPSCBC2_8X_VL/HL Pad is shown in [Table 3-4](#). Input function C = PAD&IE.

Table 3-4 Truth Table of SPSCBC2_8X_VL/HL Pad

Function	A	PAD	C	IE	OE	WPU	WPUS	WPDO	WPDI
Input	X	0	0	1	0	0	X	0	0
	X	1	1	1	0	0	X	0	0
Output	0	0	0	0	1	0	X	0	0
	1	1	0	0	1	0	X	0	0
Inout	0	0	0	1	1	0	X	0	0
	1	1	1	1	1	0	X	0	0
Tri State	X	HZ	0	0	0	0	X	0	0
	X	HZ	0	0	0	0	X	0	0
Pull up	X	1	0	0	0	WPU/WPUS: [00, 01, 10, 11]=[0, 4.7K, 20K, 1.8K]		0	0
Pull Down	X	0	0	0	0	0	X	1	0
IO Power-down	X	X	1	X	X	X	X	X	1

Driver Strength Select Function of SPSCBC2_8X_VL/HL Pad is shown in [Table 3-5](#).

VDDIO = 1.8V, All Corner (SS) @0.2 × VDDIO; 0.8 × VDDIO.

Table 3-5 Driver Strength Select Function of SPSCBC2_8X_VL/HL Pad

DRV[1]	DRV[0]	Driver Current
0	0	2mA
0	1	4mA
1	0	6mA
1	1	8mA

DC Parameters of SPSCBC2_8X_VL/HL Pad are shown in [Table 3-6](#). VDDIO = 1.8V.

Table 3-6 DC Parameter of SPSCBC2_8X_VL/HL Pad

Parameter	Description	SS	TT	FF
VT+	Schmitt trig. Low to High threshold point(1.8V)	X	>1.0V	X
VT-	Schmitt trig. High to low threshold point(1.8V)	X	>0.8V	X
RPU	Pull-up resistor(1.8V)	20k/4.7k/1.8k ±50%	20k/4.7k/1.8k ±50%	20k/4.7k/1.8k ±50%
RPD	Pull-down resistor(1.8V)	50k ± 70%	50k ± 70%	50k ± 70%
Ist	Standby Current	<2μA	<2μA	<2μA
IoZ	Tri-state output leakage current @Vo=3V or 0V	<2μA	<2μA	<2μA

AC Parameters of SPSCBC2_8X_VL/HL Pad are shown in [Table 3-7](#).

The data simulation is SS corner, Cloding = 0.3pF, Cpad = 30pF/15pF, VDDIO = 1.8V, Temp = 125°C.

Table 3-7 AC Parameter of SPSCBC2_8X_VL/HL Pad

Delay Path		Rise Time (ns)	Fall Time (ns)	Rise Delay (ns)	Fall Delay (ns)
PAD->C		0.193	0.198	0.437	0.327
A-PAD	2mA	4.911	4.766	3.379	3.54
	4mA	2.569	2.491	2.452	2.51
	6mA	1.829	1.778	2.168	2.193
	8mA	2.609	2.548	2.491	2.521

3.9.2 SPSCBC2_8X_SR_VL/HL

Base on the SPSCBC2_8X_VL/HL, improved the slew rate performance.

Truth Table of SPSCBC2_8X_SR_VL/HL Pad is shown in [Table 3-8](#). Input function C=PAD&IE.

Table 3-8 Truth Table of SPSCBC2_8X_SR_VL/HL Pad

Function	A	PAD	C	IE	OE	WPU	WPUS	WPDO	WPDI
Input	X	0	0	1	0	0	X	0	0
	X	1	1	1	0	0	X	0	0
Output	0	0	0	0	1	0	X	0	0

Function	A	PAD	C	IE	OE	WPU	WPUS	WPDO	WPDI
	1	1	0	0	1	0	X	0	0
Inout	0	0	0	1	1	0	X	0	0
	1	1	1	1	1	0	X	0	0
Tri State	X	HZ	0	0	0	0	X	0	0
	X	HZ	0	0	0	0	X	0	0
Pull up	X	1	0	0	0	WPU/WPUS: [00, 01, 10, 11]=[0, 4.7K, 20K, 1.8K]		0	0
Pull Down	X	0	0	0	0	0	X	1	0
IO Power-down	X	X	1	X	X	X	X	X	1

Driver Strength Select Function of SPSCBC2_8X_SR_VL/HL Pad is shown in [Table 3-9](#).

VDDIO = 1.8V, All Corner (SS) @0.2 × VDDIO; 0.8 × VDDIO.

Table 3-9 Driver Strength Select Function of SPSCBC2_8X_SR_VL/HL Pad

DRV[1]	DRV[0]	Driver Current
0	0	2mA
0	1	4mA
1	0	6mA
1	1	8mA

Table 3-10 DC Parameter of SPSCBC2_8X_SR_VL/HL Pad

Parameter	Description	SS	TT	FF
VT+	Schmitt trig. Low to High threshold point(1.8V)	X	1.20V	X
VT-	Schmitt trig. High to low threshold point(1.8V)	X	0.64V	X
RPU	Pull-up resistor(1.8V)		WPU/WPUS: [00, 01, 10, 11]=[0, 4.7K, 20K, 1.8K]	
RPD	Pull-down resistor(1.8V)	33K	50K	66K
Iavg	Active Current@100MHz,FF Corner,30pF loading	X	X	4mA
Ist	Standby Current	X	X	<0.1μA

Parameter	Description	SS	TT	FF
IoZ	Tri-state output leakage current @Vo=3V or 0V	X	X	<0.1μA

AC Parameters of SPSCBC2_8X_SR_VL/HL Pad are shown in [Table 3-11](#).

The data simulation is SS corner, Cloding = 0.3pF, Cpad = 30pF/15pF, VDDIO = 1.8V, Temp = 125°C.

Table 3-11 AC Parameter of SPSCBC2_8X_SR_VL/HL Pad

Delay Path		Rise Time(ns)	Fall Time(ns)	Rise Delay(ns)	Fall Delay(ns)
PAD->C		0.193	0.198	0.437	0.327
A-PAD	2mA	4.911	4.766	3.379	3.54
	4mA	2.569	2.491	2.452	2.51
	6mA	1.829	1.778	2.168	2.193
	8mA	2.609	2.548	2.491	2.521

3.9.3 SPSCBC2_8X_W_VL/HL

Base on the SPSCBC2_8X_VL/HL, enlarge the ESD area for Keypad and etc 8KV ESD performance.

Truth Table of SPSCBC2_8X_W_VL/HL Pad is shown in [Table 3-12](#). Input function C = PAD&IE

Table 3-12 Truth Table of SPSCBC2_8X_W_VL/HL Pad

Function	A	PAD	C	IE	OE	WPU	WPUS	WPDO	WPDI
Input	X	0	0	1	0	0	X	0	0
	X	1	1	1	0	0	X	0	0
Output	0	0	0	0	1	0	X	0	0
	1	1	0	0	1	0	X	0	0
Inout	0	0	0	1	1	0	X	0	0
	1	1	1	1	1	0	X	0	0
Tri State	X	HZ	0	0	0	0	X	0	0
	X	HZ	0	0	0	0	X	0	0
Pull up	X	1	0	0	0	Soft Pull-Up select WPU/WPUS: [00, 01, 10, 11]=[0, 4.7K, 20K, 1.8K]		0	0
Pull Down	X	0	0	0	0	0	X	1	0
IO Power-down	X	X	1	X	X	X	X	X	1

Driver Strength Select Function of SPSCBC2_8X_W_VL/HL Pad is shown in [Table 3-13](#).

VDDIO = 1.8V, All Corner (SS) @0.2 × VDDIO; 0.8 × VDDIO.

Table 3-13 Driver Strength Select Function of SPSCBC2_8X_W_VL/HL Pad

DRV[1]	DRV[0]	Driver Current
0	0	2mA
0	1	4mA
1	0	6mA
1	1	8mA

Table 3-14 DC Parameter of SPSCBC2_8X_W_VL/HL Pad

Parameter	Description	SS	TT	FF
V _{T+}	Schmitt trig. Low to High threshold point(1.2V)	X	0.80V	X
V _{T-}	Schmitt trig. High to low threshold point(1.2V)	X	0.39V	X
V _{T+}	Schmitt trig. Low to High threshold point(1.8V)	X	1.20V	X
V _{T-}	Schmitt trig. High to low threshold point(1.8V)	X	0.64V	X
RPU	Pull-up resistor(1.8V)		WPU/WPUS: [00, 01, 10, 11]=[0, 4.7K, 20K, 1.8K]	
RPD	Pull-down resistor(1.8V)	33K	50K	66K
Iavg	Active Current@100MHz,FF Corner,30pF loading	X	X	4mA
Ist	Standby Current	X	X	<0.1μA
IoZ	Tri-state output leakage current @Vo=3V or 0V	X	X	<0.1μA

AC Parameters of SPSCBC2_8X_W_VL/HL Pad are shown in [Table 3-15](#).

The data simulation is SS corner, Cloding = 0.3pF, Cpad = 15pF, VDDIO = 1.8V, Temp = 125°C.

Table 3-15 AC Parameter of SPSCBC2_8X_W_VL/HL Pad

Delay Path		Rise Time(ns)	Fall Time(ns)	Rise Delay(ns)	Fall Delay(ns)
PAD->C		0.002	0.0014	0.118	0.084
A-PAD	100Ω/9mA	1.45	1.54	0.21	0.144
	66Ω/13mA	1.01	1.24	0.2	0.134
	50Ω/18mA	0.74	0.89	0.187	0.127
	40Ω/23mA	0.658	0.72	0.17	0.125

Delay Path		Rise Time(ns)	Fall Time(ns)	Rise Delay(ns)	Fall Delay(ns)
	33Ω/27mA	0.57	0.63	0.164	0.141
	20Ω/45mA	0.344	0.356	0.193	0.155

3.9.4 SPSEBC2_24X_VL/HL

SPSEBC2_24X_VL/HL IO PAD is design for eMMC. The function is a managed memory capable of storing code and data. It is specifically designed for mobile devices. The eMMC is intended to offer the performance and features required by mobile devices while maintaining low power consumption. The eMMC device contains features that support high throughput for large data transfers and performance for small random data more commonly found in code usage. It also contains many security features.

Truth Table of SPSEBC2_24X_VL/HL Pad is shown in [Table 3-16](#). Input function C = PAD&IE.

Table 3-16 Truth Table of SPSEBC2_24X_VL/HL Pad

Function	A	PAD	C	IE	OE	WPU	WPUS	WPDO	WPDI
Input	X	0	0	1	0	0	X	0	0
	X	1	1	1	0	0	X	0	0
Output	0	0	0	0	1	0	X	0	0
	1	1	0	0	1	0	X	0	0
Inout	0	0	0	1	1	0	X	0	0
	1	1	1	1	1	0	X	0	0
Tri State	X	HZ	0	0	0	0	X	0	0
	X	HZ	0	0	0	0	X	0	0
Pull up	X	1	0	0	0	Soft Pull-Up select WPU/WPUS: [00, 01, 10, 11]=[0, 4.7K, 20K, 1.8K]		0	0
Pull Down	X	0	0	0	0	0	X	1	0
IO Power-down	X	X	1	X	X	X	X	X	1

Table 3-17 Driver Strength Select Function of SPSEBC2_24X_VL/HL Pad

DRV[2]	DRV[1]	DRV[0]	Driver Strength I
0	0	0	5mA
0	0	1	10mA
0	1	0	15mA
0	1	1	20mA

DRV[2]	DRV[1]	DRV[0]	Driver Strength I
1	0	0	25mA
1	0	1	30mA
1	1	0	35mA
1	1	1	40mA

Table 3-18 DC Parameter of SPSEBC2_24X_VL/HL Pad

Parameter	Description	SS	TT	FF
VT+	Schmitt trig. Low to High threshold point(1.8V)	X	>1.0V	X
VT-	Schmitt trig. High to low threshold point(1.8V)	X	>0.8V	X
RPU	Pull-up resistor(1.8V)	20K/4.7K/1.5K $\pm$ 50%	20K/4.7K/1.5K $\pm$ 50%	20K/4.7K/1.5K $\pm$ 50%
RPD	Pull-down resistor(1.8V)	50k $\pm$ 70%	50k $\pm$ 70%	50k $\pm$ 70%
Ist	Standby Current	<2 μ A	<2 μ A	<2 μ A
IoZ	Tri-state output leakage current @Vo=1.8V or 0V	<2 μ A	<2 μ A	<2 μ A

AC Parameters of SPSEBC2_24X_VL/HL Pad are shown in [Table 3-19](#).

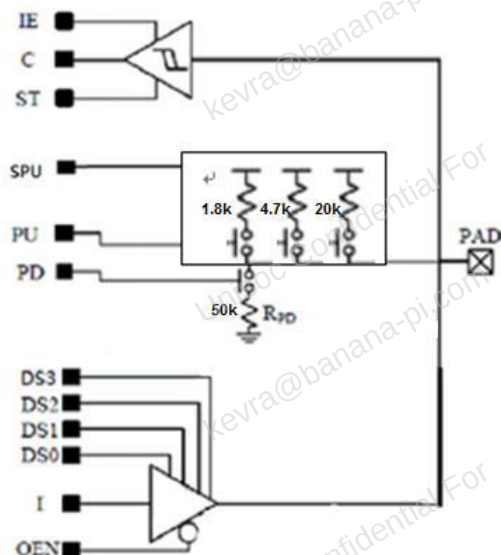
The data simulation is SS corner, clk = 208MHz, Cloding = 0.2pF, VDDIO = 1.8V, Temp = 125°C.

Table 3-19 AC Parameter of SPSEBC2_24X_VL/HL Pad

Delay Path (mA)		Raising Time (ns)	Fall Time (ns)	Raising Delay (ns)	Fall Delay (ns)
PAD -> C		0.51	0.736	0.162	0.153
A-> PAD	5	5.594	14.547	9.386	25.06
	10	3.031	6.634	3.774	10.637
	15	2.365	4.457	2.315	6.246
	20	1.976	3.443	1.666	4.31
	25	2.029	2.956	1.349	3.275
	30	1.83	2.57	1.131	2.631
	35	1.728	2.311	0.969	2.198
	40	1.615	2.111	0.858	1.886

3.9.5 PDWUWSWCDGSDE_V/H

Figure 3-2 Diagram of PDWUWSWCDGSDE_V/H Pad



Truth Table of PDWUWSWCDGSDE_V/H Pad is shown in [Table 3-20](#).

Input function $C = PAD \& IE$, Output function: $PAD = A \& OE$.

Table 3-20 Truth Table of PDWUWSWCDGSDE_V/H Pad

INPUT													OUTPUT		Remark
MS	OEN	I	DS0	DS1	DS2	DS3	PU	SPU	PD	ST	IE	PAD	PAD	C	
1	0	0	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	-	0	0	1.8V UD operation
1	0	1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0	-	1	0	
1	0	1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	1	-	1	1	
0	0	0	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	-	0	0	3.0V OD operation
0	0	1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0	-	1	0	
0	0	1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	1	-	1	1	
0/1	1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0	-	0	Input mode
0/1	1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0	1	-	0	
0/1	1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	1	1	-	1	
0/1	1	0/1	0/1	0/1	0/1	0/1	0	0	0	0/1	0	Z	-	0	
0/1	1	0/1	0/1	0/1	0/1	0/1	0	0	0	0/1	1	Z	-	X	
0/1	1	0/1	0/1	0/1	0/1	0/1	1	0	1	0/1	0/1	N.A.	-	N.A.	Forbidden
0/1	1	0/1	0/1	0/1	0/1	0/1	0	1	1	0/1	0/1		-		
0/1	1	0/1	0/1	0/1	0/1	0/1	1	0	0	0/1	0	Z	H	0	20k weak pull
0/1	1	0/1	0/1	0/1	0/1	0/1	1	0	0	0/1	1	Z	H	1	
0/1	1	0/1	0/1	0/1	0/1	0/1	0	1	0	0/1	0	Z	H	0	4.7k strong pull
0/1	1	0/1	0/1	0/1	0/1	0/1	0	1	0	0/1	1	Z	H	1	
0/1	1	0/1	0/1	0/1	0/1	0/1	1	1	0	0/1	0	Z	H	0	1.8k strong pull
0/1	1	0/1	0/1	0/1	0/1	0/1	1	1	0	0/1	1	Z	H	1	
0/1	1	0/1	0/1	0/1	0/1	0/1	0	0	1	0/1	0/1	Z	L	0	Pull down

Driver Strength Select Function of PDWUWSWCDGSDE_V/H Pad is shown in [Table 3-21](#).

1.8V mode: All Corner (SS) @0.5 × VDDIO.

Table 3-21 Driver Strength Select Function of PDWUWSWCDGSDE_V/H Pad

DS3	DS2	DS1	DS0	Driver Current
0	0	0	0	2.4mA
0	0	0	1	5.5mA
0	0	1	0	7.5mA
0	0	1	1	10.4mA
0	1	0	0	12.4mA
0	1	0	1	15mA
0	1	1	0	17.4mA
0	1	1	1	20mA
1	0	0	0	24.6mA
1	0	0	1	27.8mA
1	0	1	0	29.6mA
1	0	1	1	32mA
1	1	0	0	34.2mA
1	1	0	1	37mA
1	1	1	0	39mA
1	1	1	1	42mA

3.9.6 SPSEBC2_SWP_HL/VL

SPSEBC2_SWP_VL/HL IO PAD is design for eSE SWP pin. The IO character is the same with SPSEBC2_4X_VL/HL IO, the only difference is the PD resistor, SWP IO is 2.25K (±20%) for 800μA current detect and 24X IO is 50K (±70%).

- 1.62V ~ 1.98V VDDIO supply operation
- 0.72V ~ 0.88V VDD supply operation
- Operating junction temperature range: -40°C ~ 125°C
- Single wire protocol Bi-direction Pad support Voltage input or Voltage output or Current output.
- Minimum Signal bit duration: 1μs.
- Total area(width × height): 60μm × 335μm

Pin List

Table 3-22 SWPIO pin list1

Pin Name	Voltage Level	Description
VDD	0.75V	Core Power Supply.
VSS	0V	Core Ground Supply.
VDDIO	1.875V	IO post driver Power Supply.
VSSIO	0V	IO post driver Ground Supply.

Table 3-23 SWPIO pin list2

Pin Name	Default	Description
WPDO		SWP S2 current signal output control: 1'b0 current output=low(0μA) 1'b1 current output=high(800μA)
WPDI	1'b0	<i>If it is not used, the pin is floating.</i>
WPU	1'b0	Soft Pull-Up Enable, Active=1
WPUS	1'b0	Soft Pull-Up select signal.0 for big res, and 1 for small res.
A		Input pin from digital to SWPIO PAD
PAD		Bonding PAD pin
PADI		PAD signal after the second protection resistor. Can be used for analog signal.
C		Output pin to digital from SWPIO PAD
DRV[2]	1'b0	Driver Strength Select pin, bit 2
DRV[1]	1'b0	Driver Strength Select pin, bit 1
DRV[0]	1'b0	Driver Strength Select pin, bit 0
OE	1'b0	Output Enable, Active=1. Single wire protocol do not need voltage output function, it can be used in debug mode.
IE	1'b1	Input Enable, Active=1
SE	1'b1	Schmitt trigger Input Enable, Active=1
POC	1'b0	SWPIO output isolation when VDD is powered off, Active=1 tie SWPIO output to high/low; POC=0 SWPIO work in normal mode.(POC come from SPVDD2POC_VL_base)

Pin Name	Default	Description
SWP_BG_ITRIM[3:0]	4'b0111	trim SWP output current: 4'b0000 => -46.6% 4'b0001 => -40% ... 4'b0110 => -6.7% 4'b0111(default) => + 0% 4'b1000 => + 6.7% ... 4'b1110 => + 46.6% 4'b1111 => + 53.4%
SWP_BG_RBIAS_MODE	1'b0	SWP BG rbias mode: 1'b0 turn off rbias mode(default); 1'b1 turn on rbias mode;
SWP_BG_TEST_EN	1'b0	SWP BG output reference test path: 1'b0 turn off output reference test path(default); 1'b1 turn on output reference test path;
SWP_PG_PD	1'b0	SWP BG power up/down: 1'b0 power up swp bg(default); 1'b1 power down swp bg;

Function Description

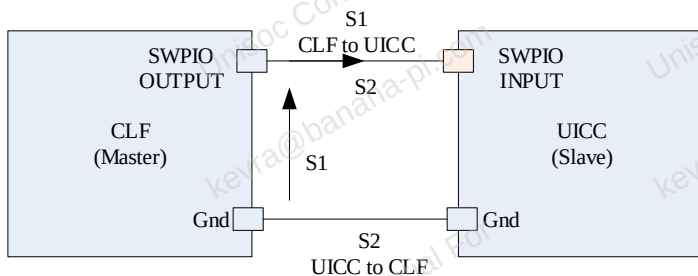
Table 3-24 SWPIO cell list

Name	Area	Description
SPSCBC2_SWP_VL/HL	60μm × 335μm	Bi-direction Pad with Enable Controlled Pull-Up Resistor. Programmable Driver Strength 2mA/4mA/6mA/8mA. No tolerance function. The Schmitt trigger input function can be controlled by the SE signal. SE=1, Schmitt input enable. SE=0 Schmitt input disable. Support 800μA current output controlled by WPDO.

Principle of the Single Wire Protocol

The SWP interface is a bit oriented, point to point communication protocol between a UICC and a contactless front-end(CLF) as shown in figure below.

Figure 3-3 SWP data transmission



The principle of the Single Wire Protocol is based on the transmission of digital information in full duplex mode:

1. The signal S1 is transmitted by a digital modulation (L or H) in the voltage domain.
2. The signal S2 is transmitted by a digital modulation (L or H) in the current domain.

Electrical Specifications

Table 3-25 SWPIO electrical Specifications

Item	Parameter	Specification			Unit
		Min	Typical	Max	
VDDIO	SWPIO supply (Class C)	1.62		1.98	V
VDD	Digital supply	0.72		0.88	V
Icc	Average current over 1ms			5	mA
Temperature	Junction temperature	-40	30	125	
Area	IP in die 60μm × 335μm		0.02		mm ²
V _{OH}	Master SWP S1 (PAD) output high voltage when $I_{Lmin} \leq S2 \leq I_{Hmax}$ (Class C)	1.377		1.98	V
V _{OL}	Master SWP S1 (PAD) output Low voltage when $-20\mu A \leq S2 \leq 0\mu A$ (Class C)	0		0.297	V
V _{IH}	Slave SWP S1(PAD) input high voltage (Class C)	1.134		2.28	V
V _{IL}	Slave SWP S1(PAD) input low voltage (Class C)	-0.3		0.495	V
I _H	Slave SWP S2(PAD) output high current when $V_{IHmin} \leq S1 \leq I_{Hmax}$	600		1000	μA
I _L	Slave SWP S2(PAD) output low current when $V_{IHmin} \leq S1 \leq I_{Hmax}$	0		20	μA
tf	Fall time of S1 when $C_{load} \leq 10pF$, $T < 5\mu s$	5ns		0.05×T	
	Fall time of S1 when $C_{load} \leq 10pF$, $T > 5\mu s$	5ns		250ns	
tr	Rise time of S1 when $C_{load} \leq 10pF$, $T < 5\mu s$	5ns		0.05×T	

Item	Parameter	Specification			Unit
		Min	Typical	Max	
	Rise time of S1 when $C_{load} \leq 10pF$, $T > 5\mu s$	5ns		250ns	
T_{H1}	Duration of the state H for coding a logical 1 of S1	$0.7 \times T$	$0.75 \times T$	$0.8 \times T$	
T_{H1}	Duration of the state H for coding a logical 0 of S1	$0.2 \times T$	$0.25 \times T$	$0.3 \times T$	
T	Default bit duration	1		5	μs
	Extended bit duration	0.59		10	μs
I _q	S1 mode at $T=1\mu s$				
	S2 mode at $T=1\mu s$				

Figure 3-4 SWPIO Voltage definition for the signal S1

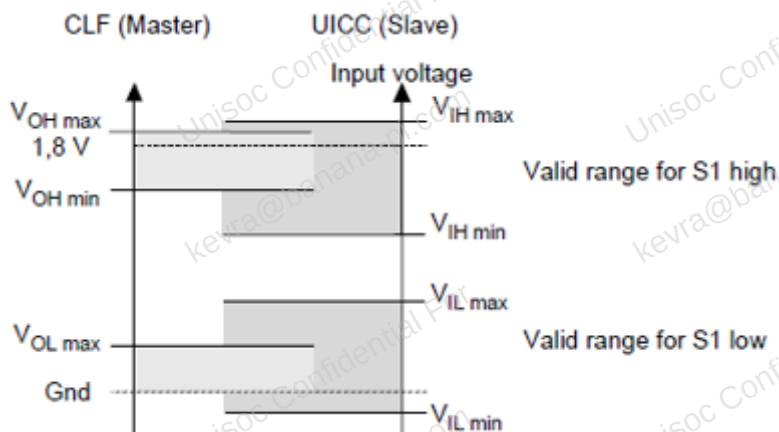
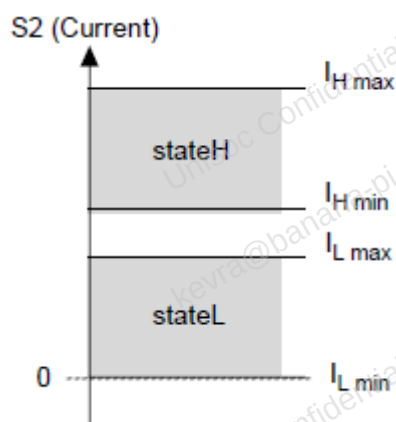


Figure 3-5 SWPIO Current definition for the signal S2



The delay path inside SWPIO: (SS corner, Cloding = 0.3pF, Cpad = 30pF/15pF, VDDIO = 1.875V, Temp = 125°C).

Table 3-26 SWPIO delay path

Delay Path		Rise Time(ns)	Fall Time(ns)	Rise Delay(ns)	Fall Delay(ns)
PAD->C		0.193	0.198	0.437	0.327
A->PAD	2mA	4.911	4.766	3.379	3.54
	4mA	2.569	2.491	2.452	2.51
	6mA	1.829	1.778	2.168	2.193
	8mA	2.609	2.548	2.491	2.521
WPDO->PAD		2.51	4.138	3.09	3.97

SPSCBC2_SWP_VL/HL setting

Bi-direction Pad with Enable Controlled Pull-Up Resistor. Programmable Driver Strength. No tolerance function. The Schmitt trigger input function can be control by the SE signal. SE = 1, Schmitt input enable. SE = 0 Schmitt input disable. Support 800μA current output controlled by WPDO.

Figure 3-6 SWPIO diagram

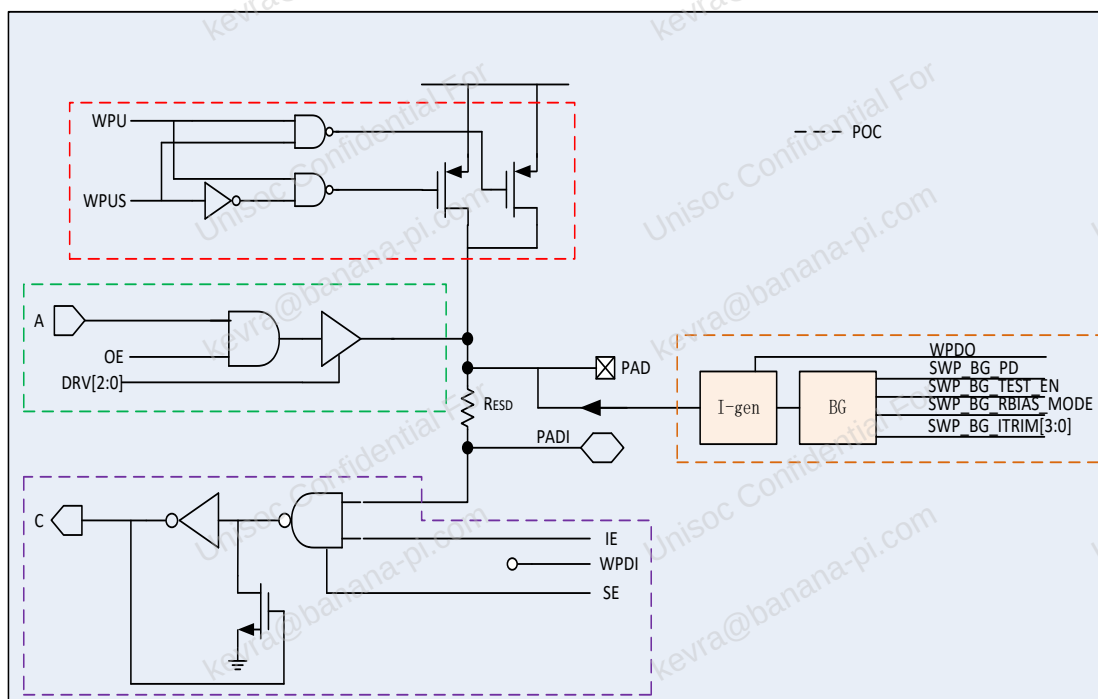
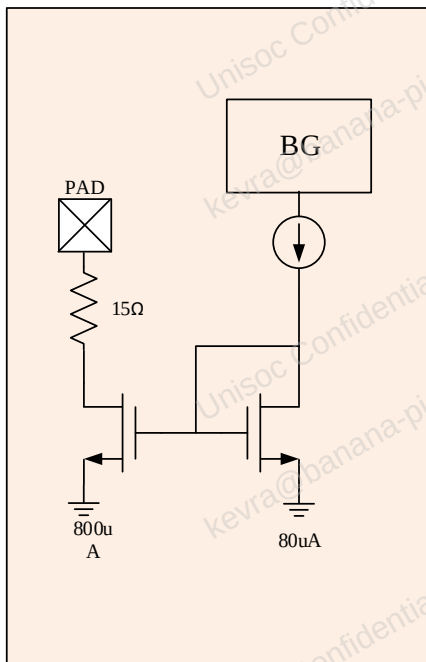


Figure 3-7 SWPIO current generation



Truth Table: Input function C = PAD&IE.

- Output function: PAD = A&OE.
- Output current function: PAD = WPDO&(~SWP_BG_PD).

Table 3-27 SWPIO application setting

Function	A	PAD	C	IE	OE	WPU	WPUS	WPDO	WPDI	SWP_PG_PD
Input voltage	X	0	0	1	0	0	X	0	0	1
	X	1	1	1	0	0	X	0	0	1
Output Voltage	0	0	0	0	1	0	X	0	0	1
	1	1	0	0	1	0	X	0	0	1
Output current	X	1(I _L)	0	0	0	0	X	0	0	0
	X	1(I _H)	0	0	0	0	X	0	1	0
Tri State	X	HZ	0	0	0	0	X	0	0	1
	X	HZ	0	0	0	0	X	0	0	1
Pull up	X	1	0	0	0	1	0: 20K(1.8V) 1:4.7K(1.8V)	0	0	1

Table 3-28 Driver Strength Select Function of SPSEBC2_SWP_HL/VL Pad

DRV[2]	DRV[1]	DRV[0]	Driver Current
0	0	0	2mA
0	0	1	4mA
0	1	0	6mA
0	1	1	8mA

4 Electrical Specification

4.1 DC Specification

4.1.1 Absolute Maximum Ratings

The functionality is subject to the absolute maximum/minimum values listed in [Table 4-1](#). Do not exceed these parameters or the part may be damaged permanently. Operation at absolute maximum ratings is not guaranteed.

Table 4-1 Absolute maximum ratings

Symbol	Parameter	Min	Max	Unit
V _{DI_18V}	Input voltage on 1.8V domain digital input(IO)	-0.3	1.98	V
V _{DI_3V}	Input voltage on 3V domain digital input(IO)	-0.3	3.3	V
V _{AI_18V}	Input voltage on 1.8V domain analog input	-0.3	1.98	V
V _{AI_12V}	Input voltage on 1.2V domain analog input	-0.3	1.3	V
V _{max, ESD}	HBM (Human Body Model) (IO)		2000	V
I _{max, DC}	Maximum DC Input current for any non-supply pin(IO)		5	mA
T _{storage}	Storage temperature	-40	+125	°C
T _c	Case temperature(SoC)	-20	+85	°C
T _{efuse}	Fuse programming temperature(SoC)	-30	+85	°C

4.1.2 Recommended Operating Conditions

The recommended operating conditions are list in [Table 4-2](#).

(Recommended voltages limits refers to die-level internal voltages)

Table 4-2 Recommended operating conditions

Symbol	Parameter	Min	Typical	Max	Unit
VDDCORE	Power supply for Core domain	0.585 0.675	0.65(deep sleep) 0.75	0.715 0.825	V
VDDCPU_LIT	Ananke supply voltage	0.585		1.045	V
VDDCPU_MID	Prometheus supply voltage	0.585		1.045	V
VDDCPU_BIG	Prometheus supply voltage	0.675		1.1	V

Symbol	Parameter	Min	Typical	Max	Unit
VDDAI	AI supply voltage	0.585		0.825	V
VDDMM	MM supply voltage	0.585		0.825	V
VDDGPU	GPU supply voltage	0.585		0.825	V
VDDMODEM	Modem supply voltage	0.585		0.825	V
VDDSRAM	SRAM supply voltage	0.675	0.75	0.825	V
VDDSRAM_CPU_LIT	Power supply for dual rail SRAM of Little core of CPU	0.675		1.045	V
VDDSRAM_CPU_MID	Power supply for dual rail SRAM of Mid core of CPU	0.675		1.045	V
AVDD1V8_BB	Baseband supply voltage	1.62	1.8	1.98	V
VDDPUB	Power supply for LPDDR4X controller	0.585		0.825	V
VDDMEM	Power supply for LPDDR4X core power	1.06	1.1	1.17	V
VDDMEMQ	Power supply for LPDDR4X IO	0.57	0.6	0.65	V
AVDD1V8_EFU	EFUSE 1.8V analog power supply	1.62	1.8	1.98	V
AVDD1V8_MPLLBIG	MPLLBIG 1.8V analog power supply	1.62	1.8	1.98	V
AVDD1V8_MPLLLIT	MPLLLIT 1.8V analog power supply	1.62	1.8	1.98	V
AVDD1V8_MPLLMID	MPLLMID 1.8V analog power supply	1.62	1.8	1.98	V
AVDD1V8_DPLL	DDR PLL supply voltage	1.62	1.8	1.98	V
AVDD1V8_PCIEPLL	PCIEPLL 1.8V analog power supply	1.62	1.8	1.98	V
AVDD1V8_USBPLL	USB3.1 PLL 1.8V analog power supply	1.62	1.8	1.98	V
AVDD1V8_DSI	MIPI DSI supply voltage	1.62	1.8	1.98	V
AVDD1V2_DSI	MIPI DSI supply voltage	1.08	1.2	1.32	V
AVDD1V8_CSI	MIPI CSI supply voltage	1.62	1.8	1.98	V
AVDD1V2_CSI	MIPI CSI supply voltage	1.08	1.2	1.32	V
AVDD1V8_LVDSRF	LVDSRF supply voltage	1.62	1.8	1.98	V
AVDD1V2_LVDSRF	LVDSRF supply voltage	1.08	1.2	1.32	V
AVDD1V2_USB	USB3.1 1.2V analog power supply	1.08	1.2	1.32	V
AVDD1V8_USB	USB PHY supply voltage	1.62	1.8	1.98	V
AVDD3V3_USB	USB PHY supply voltage	2.7	3.3	3.63	V
VDD0V75_PCIE	PCIe 0.75V digital power supply	0.675	0.75	0.825	V

Symbol	Parameter	Min	Typical	Max	Unit
VDD0V75_UFS	UFS 0.75V digital power supply	0.675	0.75	0.825	V
VDD0V75_USB	USB3.1 0.75V digital power supply	0.675	0.75	0.825	V
AVDD1V2_PCIE	PCIe 1.2V analog power supply	1.08	1.2	1.32	V
AVDD1V2_UFS	UFS 1.2V analog power supply	1.08	1.2	1.32	V
VDD1V8_ISE	ISE 1.8V digital power supply	1.62	1.8	1.98	V

4.1.3 ESD Characteristics

The ESD characteristics are shown in [Table 4-3](#).

Table 4-3 ESD characteristics

Symbol	Parameter	Condition	Value	Unit
HBM	Human body model	ANSI/ESDA/JEDEC JS-001	±2000	V
CDM	Charged-device model	ANSI/ESDA/JEDEC JS-002	±500	V
LU	Latch UP current	JESD78E	±100	mA

4.1.4 DC Characteristics

The typical core voltage (VDDCORE) is 0.75V and the digital I/O supply (VIO1V8) is typically at 1.875V. The power pins should be connected with a decoupling capacitor to ground (VSS).

For the following table, $T_{amb} = -30^{\circ}\text{C}$ to $+60^{\circ}\text{C}$, $VSS = 0\text{V}$ (ground), and all voltages are measured with respect to VSS, unless otherwise specified.

Table 4-4 DC Characteristics

Symbol	Parameter	Conditions	Min	Typical	Max	Unit
Digital supply voltage: pins VIO1V8						
VIO1V8	Digital supply voltage		1.62		1.98	V
Digital supply voltage: pins VSIM0						
VSIM0	Digital supply voltage		1.62		3.3	V
Digital supply voltage: pins VSIM1						
VSIM1	Digital supply voltage		1.62		3.3	V
Digital supply voltage: pins VSD0						
VSD0	Digital supply voltage		1.62		3.3	V
Digital supply voltage: pins VSD2						

Symbol	Parameter	Conditions	Min	Typical	Max	Unit
VSD2	Digital supply voltage		1.62		3.3	V
Digital supply voltage: pins VDDMEM/BDDMEMQ(DDR)						
VDDMEM	Digital supply voltage		1.06	1.1	1.17	V
VDDMEMQ	IO supply voltage		0.57	0.6	0.65	V
Digital supply voltage: pins VEMMCIO						
VEMMCIO	Digital supply voltage		1.62		1.98	V
Digital supply voltage: internal VDDCORE(SoC)						
VDDCORE	Digital core supply voltage		0.675	0.75	0.825	V
	Sleep mode digital core supply voltage		0.585	0.65	0.715	V
Digital input(IO)						
V _{IL}	Input voltage LOW-level		0		0.3 VDDIO	V
V _{IH}	Input voltage HIGH-level		0.7 VDDIO		VDDIO	V
I _{LI}	Input leakage current			±2		μA
Digital output(IO)						
V _{OL}	Output voltage LOW-level	At I _{sink} = 2,4,6,8 mA (programmable)	0		0.1 VDDIO	V
V _{OH}	Output voltage HIGH-level	At I _{source} = 2,4,6,8 mA (programmable)	0.9 VDDIO		VDDIO	V
Analog supply voltage						
AVDD1V8_BB	Baseband supply voltage		1.62	1.8	1.98	V
VDDPUB	Power supply for LPDDR4X controller		0.585		0.825	V
VDDMEM	Power supply for LPDDR4X core power		0.99	1.1	1.21	V
VDDMEMQ	Power supply for LPDDR4X IO		0.57	0.6	0.65	V
AVDD1V8_EFU	EFUSE 1.8V analog power supply		1.62	1.8	1.98	V
AVDD1V8_MPLLBIG	MPLLBIG 1.8V analog power supply		1.62	1.8	1.98	V

Symbol	Parameter	Conditions	Min	Typical	Max	Unit
AVDD1V8_MPLLLIT	MPLLLIT 1.8V analog power supply		1.62	1.8	1.98	V
AVDD1V8_MPLLMID	MPLLMID 1.8V analog power supply		1.62	1.8	1.98	V
AVDD1V8_DPLE	DDR pll supply voltage		1.62	1.8	1.98	V
AVDD1V8_PCIEPLL	PCIEPLL 1.8V analog power supply		1.62	1.8	1.98	V
AVDD1V8_USBPLL	USB3.1 PLL 1.8V analog power supply		1.62	1.8	1.98	V
AVDD1V8_DSI	MIPI DSI supply voltage		1.62	1.8	1.98	V
AVDD1V2_DSI	MIPI DSI supply voltage		1.08	1.2	1.32	V
AVDD1V8_CSI	MIPI CSI supply voltage		1.62	1.8	1.98	V
AVDD1V2_CSI	MIPI CSI supply voltage		1.08	1.2	1.32	V
AVDD1V8_LVDSRF	LVDSRF supply voltage		1.62	1.8	1.98	V
AVDD1V2_LVDSRF	LVDSRF supply voltage		1.08	1.2	1.32	V
AVDD1V8_USB	USB PHY supply voltage		1.7	1.8	1.98	V
AVDD3V3_USB	USB PHY supply voltage		3.0	3.3	3.6	V
VDD0V75_PCIE	PCIe 0.75V digital power supply		0.675	0.75	0.825	V
VDD0V75_UFS	UFS 0.75V digital power supply		0.675	0.75	0.825	V
VDD0V75_USB	USB3.1 0.75V digital power supply		0.675	0.75	0.825	V
AVDD1V2_PCIE	PCIe 1.2V analog power supply		1.08	1.2	1.32	V
AVDD1V2_UFS	UFS 1.2V analog power supply		1.08	1.2	1.32	V
AVDD1V2_USB	USB3.1 1.2V analog power supply		1.08	1.2	1.32	V

4.2 AC Characteristics

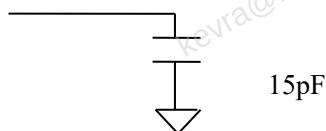
A pin's AC characteristics include input and output capacitance, which determine loading for external drivers or other load analysis. The AC characteristics also include a de-rating factor, which indicates how much faster or slower the AC timings get with different loads.

Table 4-5 Standard input, output and I/O pin AC characteristics

Symbol	Parameters	Min	Typical	Max	Unit
C_{in}	Input capacitance, all standard input and IO pins			3.5	pF
C_{load}	Output capacitance, all standard output and IO pins			30	pF
T_{dr}	Output de-rating falling edge on all standard output and I/O pins, from 15pF load		0.075		ns/pF

NOTE

The AC specifications are tested with a 15pF load as indicated

Figure 4-1 Test circuit of an I/O pin


The output capacitance and de-rating falling edge are measured under the condition of maximum driving strength: 24 mA @ 1.8 V.

For the following tables, $T_c = -20^{\circ}\text{C}$ to $+85^{\circ}\text{C}$, $V_{SS} = 0\text{ V}$ (ground), and all voltages are measured with respect to V_{SS} , unless otherwise specified.

Table 4-6 System Clock AC characteristics

Symbol	Parameter	Conditions	Min	Typical	Max	Unit
f_{mclk}	Master clock frequency			52		MHz
C_I	Input capacitance			30		pF
Master clock input						
V_{pp}	Sine wave peak-to-peak swing	Loading = 30pF	0.7		1.8	V
Real-time clock input						
f_{rtclk}	Real time clock frequency			32		kHz
$\Delta f/f_{rtclk}$	Frequency tolerance	Standard deviation	-80		80	ppm
δ_{rtclk}	Clock duty cycle		35		65	%

4.3 Performance Specifications

4.3.1 LVDSRF

The LVDS interface is designed for transmitting and receiving data between RF chip and BB chip. The recommended line rate is 1.04Gbps.

4.3.1.1 DC Specifications

Table 4-7 LVDS TX DC specifications

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Vod	Transmit Differential output voltage magnitude	190	200	210	mV
$\Delta Vod $	Change in Differential output voltage magnitude between logic states			25	mV
VCMTX	Steady-state common-mode output voltage	180	200	220	mV
$\Delta VCMTX$	Changes in steady-state common-mode output voltage between logic states			25	mV
Zos	Single-ended output impedance		50		Ω

Table 4-8 LVDS RX DC specifications

Parameter	Conditions/Comments	Min	Typical	Max	Unit
WIDTH	Differential input high voltage threshold		50		mV
VIDTL	Differential input low voltage threshold	-50			mV
VCMRX_DC	Input common mode voltage		200		mV
ZID	Differential input impedance (before calibration)	80	100	125	Ω

4.3.1.2 AC Specifications

Table 4-9 LVDS TX AC specifications

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Cnear	Near-end capacitor to ground		2	3	pF
Cfar	Near-end capacitor to ground		2	3	pF
tr	Differential output signal rise time (20%-80%)			0.1	UI
tf	Differential output signal fall time (80%-20%)			0.1	UI
$\Delta VCMTX$	Changes in common-mode output voltage when random bit stream in		50	100	mVpp

Table 4-10 LVDS RX AC specifications

Parameter	Conditions/Comments	Min	Typical	Max	Unit
$\Delta V_{CMRX}(HF)$	Common mode interference beyond 450MHz			200	mVpp
$\Delta V_{CMRX}(LF)$	Common mode interference between 50MHz and 450MHz	-50		50	mVpp
CCM	Common mode termination		20		pF

4.3.2 SOC Phase-Locked- Loop (PLL)

Table 4-11 Performance of the phase-locked loop

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Input reference	All PLLs except DeskewPLL		26		MHz
Input reference	DeskewPLL	100	500	800/1066.5	MHz
Frequency Range	PIXELPLL	1600	2673	3200	MHz
	GPLL	800	858	1800	MHz
	AUDPLL	800	1228.88	1600	MHz
	VDSPPLL	800	1014	1600	MHz
	AIPLL	800	988	1600	MHz
	PHYCPR8PLL	800	1222	1222	MHz
	CPLL	1022.36	1022.36	1022.36	MHz
	MPLL_BIG	1200	2600	3200	MHz
	MPLL_MID	1200	2392	3200	MHz
	MPLL_LIT	1200	2002	2400	MHz
	MPLL_SCU	1000	1586	2000	MHz
	PSCPR8PLL	800	1222	1222	MHz
	TGPLL	1536	1536	1536	MHz
	V4NRPLL	2457.6	2457.6	2457.6	MHz
	LVDSPLL	1.456	2.912	4	GHz
	DPLL	100	1066	3200	MHz
	DeskewPLL	400	2000	4266	MHz
Settling time				200	μ s
Sleep mode	Yes				

4.3.3 Thermal Sensor

This is a temperature-to-digital converter using an on-chip bandgap temperature sensor. The overall accuracy is $\pm 3^{\circ}\text{C}$ (typical) from 0°C to 90°C .

This is suitable for multi-core smart phone applications requiring highly accurate temperature sensing to perform thermal control.

Table 4-12 Performance of the Thermal sensor

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog supply voltage	1.62	1.8	1.98	V
	Digital core supply voltage	0.675	0.75	0.825	V
Temperature resolution			0.26		$^{\circ}\text{C}$
Temperature accuracy	$0^{\circ}\text{C} \sim 90^{\circ}\text{C}$		± 3		$^{\circ}\text{C}$
	$-20^{\circ}\text{C} \sim 0^{\circ}\text{C}$, $90^{\circ}\text{C} \sim 125^{\circ}\text{C}$		± 5		$^{\circ}\text{C}$
Conversion time		24		32	32k Clock cycle

4.3.4 USB 2.0 PHY

The USB 2.0 PHY is fully compliant with USB 2.0 specifications. It's own PLL can generate 480MHz clock for its usage. Please refer to Universal Serial Bus Specification revision 2.0 at <http://www.usb.org/developers/docs>

Table 4-13 Main performance of the USB

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog 1.8V power supply	1.7	1.8	1.98	V
	Analog 3.3V power supply	3.0	3.3	3.6	V
	Digital core supply voltage	0.675	0.75	0.825	V
THSR	Rise time(10% ~ 90%)	500			ps
THSF	Fall time (90% ~ 10%)	500			ps
ZoutDRV(HS)	Driver output impedance	40.5	45	49.5	Ω
VCRS(FS)	Crossover voltage	1.3		2.0	V
TFRFM(FS)	Rise time and fall time matching	90	100	110	%
TFR(FS)	Rise time	4		20	ns
TFF(FS)	Fall time	4		20	ns
RPU1	Bus pull_up resistor1 on upstream port	900		1575	Ω
RPU2	Bus pull_up resistor2 on upstream port	525		1515	Ω
RPD	Bus pull_down resistor on downstream port	14.25k		24.8k	Ω
VDI	Differential input sensitivity	0.2			V

Parameter	Conditions/Comments	Min	Typical	Max	Unit
VCM	Differential input common mode range	0.8		2.5	V
TFST(FS)	Width of SE0 during differential transition			14	ns
Zout DRV(FS)	Driver output resistor	28		44	Ω
VCRS(LS)	Crossover voltage	1.3		2.0	V
TFRFM(LS)	Rise time and fall time matching	80	100	120	%
TFR(LS)	Rise time	75		200	ns
TFF(LS)	Fall time	75		200	ns
VHSSQ common range	Vhssq common input range	-50		500	mv
TFST(LS)	Width of SE0 during differential transition			210	ns

4.3.5 USB3.1/DP ComboPHY

The USB3.1/DP ComboPHY is fully compliant with USB3.1/DisplayPort1.4 specifications and PIPE version4.0 interface.

Table 4-14 Main performance of the USB3.1/DP

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog 1.8V power supply	1.7	1.8	1.98	V
	Analog low power supply	0.7	0.75	0.825	V
	Digital core supply voltage	0.7	0.75	0.825	V
Differential Insertion Loss	TX Differential Routes, at baudrate/2		-1.5		dB
Differential Return Loss	TX Differential Routes, at baudrate		-10		dB
Common Mode Return Loss	TX Differential Routes, at baudrate		-6		dB
Common Mode to differential loss	TX Differential Routes, at baudrate		-20		dB
Differential to common mode conversion	TX Differential Routes, at baudrate		-20		dB
Xtalk coupling(all attackers)	TX Differential Routes, at baudrate/2		-30		dB
Differential Impedance		90	100	110	Ω
Single Ended Impedance		45	50	55	Ω
Length variation (intra differential pair)	intra differential pair		± 0.250		mm
Length variation (between lanes)	between lanes		± 0.5		mm

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Dj _{REFCLK} USB3.0	Reference clock deterministic jitter		12.0		ps
Dj _{REFCLK} USB3.1	Reference clock deterministic jitter		7.4		ps
Dj _{REFCLK} DP HBR2	Reference clock deterministic jitter		14.9		ps
Dj _{REFCLK} DP HBR3	Reference clock deterministic jitter		14.9		ps

4.3.6 MIPI CSI

MIPI CSI is fully compliant with MIPI CSI-2 V1.2 and D-PHY 1.2. Please refer specification at <http://www.mipi.org>

Table 4-15 Main performance of the MIPI CSI

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog 1.8V power supply	1.7	1.8	1.98	V
	Analog 1.2V power supply	1.1	1.2	1.3	V
	Digital core supply voltage	0.675	0.75	0.825	V
HS RX mode					
VCMRX	Common-mode voltage HS receiver mode	70		330	mV
Vidth	Differential input high threshold			70	mV
Vidtl	Differential input low threshold	-70			mV
Vihhs	Single-ended input high voltage			460	mV
Vilhs	Single-ended input low voltage	-40			mV
Zid	Differential input impedance	80		125	Ω
ΔVCMRX (HF)	Common-mode interference beyond 450MHz			100	mV
ΔVCMRX (LF)	Common-mode interference beyond 50~450MHz	-50		50	mV
LP RX mode					
VIH	Logic high	880			mV
VIL	Logic low			550	mV
VIL-ULPS	Logic low @ULP			300	mV
VHYST	Input hysteresis	25			mV
eSPIKE	Input pulse rejection			300	V•ps

Parameter	Conditions/Comments	Min	Typical	Max	Unit
TMIN-RX	Minimum pulse width response	50			ns
Data-Clock Timing Spec					
Tsetup(rx) @ 1.5Gbps		0.2			UI
Tsetup(rx) @ 1Gbps		0.15			UI
Thold(rx) @ 1.5Gbps		0.2			UI
Thold(rx) @ 1Gbps		0.15			UI

4.3.7 MIPI C-PHY CSI

MIPI C-PHY CSI is fully compliant with MIPI D-PHY V1.2 and C-PHY V1.2. Please refer specification at <http://www.mipi.org>

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog 1.8V power supply	1.7	1.8	1.98	V
	Analog 1.2V power supply	1.1	1.2	1.3	V
	Digital core supply voltage	0.675	0.75	0.825	V
HS RX mode					
VCMRX	Common-mode voltage HS receiver mode	95		390	mV
Width	Differential input high threshold			40	mV
Vidtl	Differential input low threshold	-40			mV
Vihhs	Single-ended input high voltage			535	mV
Vilhs	Single-ended input low voltage	-40			mV
Zid_ab Zid_bc Zid_ca	Differential input impedance	80	100	120	Ω
ΔZid	Differential input impedance mismatch			10	%
$\Delta VCMRX(HF)$	Common-mode interference beyond 450MHz			50	mV
$\Delta VCMRX(LF)$	Common-mode interference beyond 50~450MHz	-25		25	mV
LP RX mode					
VIH	Logic high	740			mV
VIL	Logic low			550	mV
VIL-ULPS	Logic low @ULP			300	mV
VHYST	Input hysteresis	25			mV
eSPIKE	Input pulse rejection			300	V $\cdot$ ps

Parameter	Conditions/Comments	Min	Typical	Max	Unit
TMIN-RX	Minimum pulse width response	20			ns
Timing Spec					
Receive timing budget		0.5			UI

4.3.8 MIPI DSI

MIPI DSI is fully compliant with MIPI DSI V1.2 and D-PHY 1.2. Please refer specification at <http://www.mipi.org>

Table 4-16 Main performance of the MIPI DSI

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog 1.8V power supply	1.7	1.8	1.98	V
	Analog 1.2V power supply	1.1	1.2	1.3	V
	Digital core supply voltage	0.675	0.75	0.825	V
HSTX mode					
Zos	Single ended output impedance	40	50	62.5	Ω
VCMTX	HS transmit static common-mode voltage	150	200	250	mV
$ \Delta VCMTX(1,0) $	VCMTX mismatch when output is Differential-1 or 0			5	mV
$ VOD $	HS transmit differential voltage	140	200	270	mV
$ \Delta VOD $	VOD mismatch when output is Differential-1 or 0			10	mV
VOHHS	HS output high voltage			360	mV
$\Delta VCMTX(HF)$	Common-level variations above 450MHz(RMS)			15	mV
$\Delta VCMTX(LF)$	Common-level variation between 50 ~450MHz(RMS)			25	mV
tr / tf	20% ~ 80% rise time and fall time(<1.5Gbps)	100p		0.35UI	ps
LPTX mode					
VOH	Thevenin output high level	1.1	1.2	1.3	V
VOL	Thevenin output low level	-50		50	mV
ZOLP	Output impedance of LP transmitter	110			Ω
$\Delta ZOLP(01,10)$	Single-ended output impedance mismatch driving opposite level			20	%
$\Delta ZOLP(00,11)$	Single-ended output impedance mismatch driving same level			5	%
Slew rate	Clload = 70pF	30		120	mV/ns
LPRX mode					

Parameter	Conditions/Comments	Min	Typical	Max	Unit
VIH	Logic high	740			mV
VIL	Logic low			550	mV
VHYST	Input hysteresis	25			mV
LPCD mode					
VIHCD	Logic high contention threshold	450			mV
VILCD	Logic low contention threshold			200	mV

4.3.9 MIPI MPHY4.1

MIPI MPHY is fully compliant with MIPI MPHY 4.1. Please refer specification at <http://www.mipi.org>

Table 4-17 Main performance of the MIPI MPHY

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog 1.8V power supply	1.7	1.8	1.98	V
	Analog 1.2V power supply	1.08	1.2	1.32	V
	Digital core supply voltage	0.72	0.8	0.88	V
VDIF_AC_LA_RT_TX	Large differential TX AC voltage amplitude in terminated case	140		250	mV
VDIF_DC_LA_NT_TX	Large differential TX DC voltage amplitude in unterminated case	320		480	mV
VDIF_AC_LA_NT_TX	Large differential TX AC voltage amplitude in unterminated case	280		500	mV
VDIF_DC_SA_RT_TX	Small differential TX DC voltage amplitude in terminated case	100		130	mV
VDIF_AC_SA_RT_TX	Small differential TX AC voltage amplitude in terminated case	80		140	mV
VDIF_DC_SA_NT_TX	Small differential TX DC voltage amplitude in unterminated case	200		260	mV
VDIF_AC_SA_NT_TX	Small differential TX AC voltage amplitude in unterminated case	160		280	mV
VCM_LA_TX	TX common mode voltage for large amplitude	160		260	mV
VCM_SA_TX	TX common mode voltage for small amplitude	80		190	mV
R_SE_TX	Single-ended transmitter output resistance, fixed calibration	40		60	Ω
HS TX mode					
TF_TX	Fall time	0.1			UI

Parameter	Conditions/Comments	Min	Typical	Max	Unit
TR_TX	Rise time	0.1			UI
TEYE_TX	Transmitter eye opening(HS-G1 and HS-G2)	0.2			UI
TEYE_HS_G3_TX	Transmitter eye opening(HS-G3)	0.55			UI
TSKEW_TX	Intra-differential skew	-0.06		0.06	UI
TPULSE_TX	Transmitter pulse width(HS-G1 for VDIF_DC_SA_RT_TX)	0.9			UI
ΔRSE_TX	Output resistance mismatch	-6		6	Ω
DJTX	Transmitter deterministic jitter			0.15	UI
TJTX	Transmitter total jitter			0.32	UI
STTJTX	Transmitter short term total jitter.			0.2	UI
fOFFSET_TX	Transmitter frequency offset	-2000		2000	ppm
LP TX mode					
TR_LS_TX	Rise time LS			0.2	UI_LS
TF_LS_TX	Fall time LS			0.2	UI_LS
Receiver Part					
Vdif_rt_rx	Differential RX voltage amplitude with termination	60		245	mV
Vdif_nt_rx	Differential RX voltage amplitude in untermination	120		490	mV
Vdif_acc_rx	Accumulated differential RX voltage amplitude for HS-GEAR1	40			mV
Vcm_rx	RX common-mode voltage	25		330	mV
R_Diff_RX	Differential input resistance	80		110	Ω
Teye_rx	High-speed Receiver eye opening	0.2			UI
Tpulse_rx	High-speed Receiver pulse width	0.8			UI
SJrx	High-speed Receiver sinusoidal jitter tolerance			0.35	UI
STSJrx	High-speed Receiver short term sinusoidal jitter tolerance			0.2	UI
TJrx	High-speed Receiver total jitter tolerance			0.52	UI
STTJrx	High-speed Receiver short term total jitter tolerance			0.3	UI
Vsq	Squelch exit voltage	50		140	mV

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Vdif_sq	Squelch differential voltage			20	mV
Vcm_sq	Squelch common-mode voltage	0		330	mV
Tpulse_sq	Noise pulse width			20	ns
Tspace_sq	Noise pulse spacing	500			ns
Vint_sq	Peak interfere amplitude			200	mV
Fint_sq	Interfere frequency	500			MHz

4.3.10 PCIEGEN3

Table 4-18 Main performance of the PCIEGEN3 2.5GT/s

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog supply voltage	1.7	1.8	1.98	V
	Digital core supply voltage	0.72	0.8	0.88	V
UI	unit interval	400.002	400.004	400.005	ps
VTX-DIFF-PP	Differential p-p Tx voltage swing for full swing operation	810	868	897	mV
VTX-DIFF-PP-LOW	Differential p-p Tx voltage swing for low swing operation	417	449	467	mV
TTX-UTJ	Tx uncorrelated total jitter	8.74	10.3	14.7	ps
TTX-UDJDD	Tx uncorrelated deterministic jitter	1.67	2.56	6.36	ps
TTX-UTJ-SRIS	Tx Uncorrelated Total Jitter SRIS	16.6	20.1	21.7	ps
TX-DETECT-RX	TX detect RX functionality 1-Pass, 0-Fail	1.000	1.000	1.000	MHz
BWTX-PKG-PLL1	Tx PLL BW corresponding to PKG-Tx-PLL1	8.12	11.6	14.1	MHz
PKG-TX-PLL1	Tx PLL peaking corresponding to BWTX-PKG-PLL1	0.217	0.391	0.526	dB
VRX-IDLE-DET-DIFF-PP	Electrical idle detect threshold	105	118	138	mV

Table 4-19 Main performance of the PCIEGEN3 5GT/s

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog supply voltage	1.7	1.8	1.98	V
	Digital core supply voltage	0.72	0.8	0.88	V

Parameter	Conditions/Comments	Min	Typical	Max	Unit
UI	unit interval	200.001	200.002	200.003	ps
VTX-DIFF-PP	Differential p-p Tx voltage swing for full swing operation	813	863	895	mV
VTX-DIFF-PP-LOW	Differential p-p Tx voltage swing for low swing operation	420	450	466	mV
TTX-UTJ	Tx uncorrelated total jitter	7.41	8.86	14.1	ps
TTX-UDJDD	Tx uncorrelated deterministic jitter	0.939	1.91	3.21	ps
TTX-UPW-TJ	Total Uncorrelated Pulse Width Jitter (ps PP)	5.21	7.75	10.9	ps
TTX-UPW-DJDD	Deterministic DjDD Uncorrelated Pulse Width Jitter (ps PP)	1.20	2.80	5.16	ps
TTX-UTJ-SRIS	Tx Uncorrelated Total Jitter SRIS	7.275	8.316	10.873	ps
BWTX-PKG-PLL1	Tx PLL BW corresponding to PKG-Tx-PLL1	8.1	11.6	14.1	MHz
PKG-TX-PLL1	Tx PLL peaking corresponding to BWTX-PKG-PLL1	0.217	0.391	0.526	dB
VRX-IDLE-DET-DIFF-PP	Electrical idle detect threshold	93	108	126	mV

Table 4-20 Main performance of the PCIEGEN3 8GT/s

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog supply voltage	1.7	1.8	1.98	V
	Digital core supply voltage	0.72	0.8	0.88	V
UI	unit interval	125.001	125.001	125.002	ps
VTX-DIFF-PP	Differential p-p Tx voltage swing for full swing operation	811	861	890	mV
VTX-DIFF-PP-LOW	Differential p-p Tx voltage swing for low swing operation	413	446	464	mV
VTX-EIEOS-FS	Minimum voltage swing during EIEOS for full swing signaling	285	305	320	mV
VTX-EIEOS-RS	Minimum voltage swing during EIEOS for reduced swing signaling	276	296	309	mV
TTX-UTJ	Tx uncorrelated total jitter	5.86	6.79	8.67	ps
TTX-UDJDD	Tx uncorrelated deterministic jitter	0.309	0.780	1.31	ps
TTX-UPW-TJ	Total Uncorrelated Pulse Width Jitter (ps PP)	5.90	6.84	8.37	ps

Parameter	Conditions/Comments	Min	Typical	Max	Unit
TTX-UPW-DJDD	Deterministic DjDD Uncorrelated Pulse Width Jitter (ps PP)	0.843	1.46	2.11	ps
TTX-UTJ-SRIS	Tx Uncorrelated Total Jitter SRIS	9.894	11.685	13.496	ps
BWTX-PKG-PLL1	Tx PLL BW corresponding to PKG-Tx-PLL1	2.25	3.11	3.46	MHz
PKG-TX-PLL1	Tx PLL peaking corresponding to BWTX-PKG-PLL1	0.603	0.707	0.917	dB
VTX-BOOST-FS	maximum nominal Tx boost ratio for full swing	8.92	9.29	9.41	dB
VTX-BOOST-RS	maximum nominal Tx boost ratio for reduced swing	9.55	9.65	9.91	dB
VRX-IDLE-DET-DIFF-PP	Electrical idle detect threshold	93	108	125	mV

4.3.11 Automatic Power Control DAC

One general DAC is used to control power ramping and gain. It is a 14 bits D/A converters with a programmable-gain output driver. Special offset cancellation is applied in the DAC.

Table 4-21 Performance of the APC DAC

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog supply voltage(AVDD)	1.62	1.8	1.98	V
	Digital core supply voltage(DVDD)	0.675	0.75	0.825	V
Max Output range	PGA gain=1.3	0.03		1.3	V
	PGA gain=1.5	0.04		1.4	V
	PGA gain=2.0	0.05		1.5	V
N	Resolution of the DAC		14		Bit
Sample rate				1	MHz
PGA gain		1.3	1.5	2	
DNL	Differential nonlinearity		±1	±2	LSB
Setting time	4pF Capacitor load, full scale input, 1.5X PGA gain, bypass internal RC filter		1	2	µs
PSRR		40			dB

4.3.12 CLK26M Buffer

This CLK26M Buffer is used to generate 26MHz clock for PLL and other circuit blocks. It includes one 26M_buffer and one sine driver output of this CLK26M block can be set the sine wave.

Table 4-22 Performance of the 26MHz Buffer

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Power supply	Analog supply voltage	1.62	1.8	1.98	V
	Digital core supply voltage	0.675	0.75	0.825	V
Vpp_in	Input sine wave peak to peak amplitude		325		mV
f_26	Output Clock Frequency		26		MHz
Dutycycle	Output Clock Dutycycle	45	50	55	%
f_sine	Output sine wave Frequency		26		MHz

4.3.13 RCO

Table 4-23 Performance of the RCO

Parameter	Conditions/Comments	Min	Typical	Max	Unit
Frequency range	RCO6M	5.94	6	6.06	MHz
	RCO60M	54	60	66	MHz
	RCO100M	90	100	110	MHz
	RCO150M	135	150	165	MHz

4.3.14 DDC

The DDC is used for reduction of on-die transient voltage droop with high-performance embedded circuit.

Table 4-24 Main performance of the DDC

Parameter	Conditions/Comments	Min	Typical	Max	Unit
AVDD18	Analog power supply(clean)	1.62	1.8	1.98	V
PVDD18	Line supply voltage input	1.62	1.8	1.98	V
DVDD	Digital power supply	0.675	0.75	0.825	V
Temp	Junction temperature range	-20	25	125	°C
IAVDD	Supply current from AVDD @2GHz		20		mA
IPD_AVDD	Power down leakage current from AVDD		1		μA

Parameter	Conditions/Comments	Min	Typical	Max	Unit
IPVDD	Supply current from PVDD @2GHz		0.1		mA
IPD_PVDD	Power down leakage current from PVDD		1		μA
IDVDD	Supply current from DVDD @2GHz		1		mA
IPD_DVDD	Power down leakage current from DVDD		1		μA
Vout Sensing Range	Sensing Range	0.675	0.75	1	V
CLK_ETC	Operating frequency		2	3	GHz
Jpp	Peak to peak period jitter			5	%
Tr	Response time			1	ns
Icom_tran	Max transient compensation current			2	A
Under_shot	Maximum value			100	mV

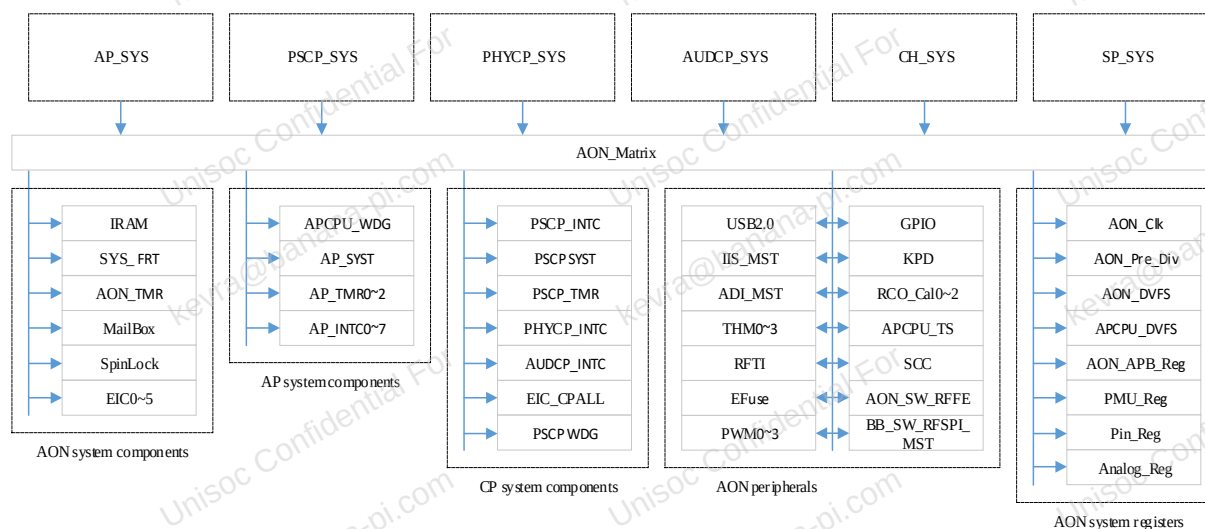
5 AON Subsystem

5.1 General Description

5.1.1 Overview

AON(Always-On) Subsystem has 88KB IRAM. Default running frequency is XTALBUF26MHz. The others options are RCO100MHz, TGPLL153MHz, TGPLL128MHz.

5.1.2 Functional Block Diagram



5.2 AON Subsystem Address Map

5.2.1 Peripheral Address Map

Start Address	End Address	Module Name	IP Name	Description
0x6000_0000	0x60FF_FFFF	PUB_CFG_Reg	PUB_CFG_Reg	PUB System configure register
0x6400_0000	0x6400_FFFF	PSCP_WDG	WDG	PSCP System Watch Dog
0x6401_0000	0x6401_FFFF	PSCP_SYST	SYST	PSCP System System Timer
0x6402_0000	0x6402_FFFF	PSCP_TMR	TMR	PSCP System General Purpose Timer
0x6403_0000	0x6403_3FFF	PWM0	PWM	Pulse Width Modulation

Start Address	End Address	Module Name	IP Name	Description
0x6403_4000	0x6403_7FFF	PWM1	PWM	Pulse Width Modulation
0x6403_8000	0x6403_BFFF	PWM2	PWM	Pulse Width Modulation
0x6403_C000	0x6403_FFFF	PWM3	PWM	Pulse Width Modulation
0x6404_0000	0x6404_FFFF	AON_TMR	TMR	AON System General Purpose Timer
0x6405_0000	0x6405_FFFF	SYS_FRT	SYS_FRT	System free run timer
0x640B_0000	0x640B_FFFF	SCC	SCC	Silicon Characterization and Calibration
0x640C_0000	0x640C_FFFF	AP_INTC0	INTC	AP System Interrupts Controller 0
0x640D_0000	0x640D_FFFF	AP_INTC1	INTC	AP System Interrupts Controller 1
0x640E_0000	0x640E_FFFF	AP_INTC2	INTC	AP System Interrupts Controller 2
0x640F_0000	0x640F_FFFF	AP_INTC3	INTC	AP System Interrupts Controller 3
0x6410_0000	0x6410_FFFF	AP_INTC4	INTC	AP System Interrupts Controller 4
0x6411_0000	0x6411_FFFF	AP_INTC5	INTC	AP System Interrupts Controller 5
0x6412_0000	0x6412_FFFF	PSCP_INTC	INTC	PSCP System Interrupts Controller
0x6413_0000	0x6413_FFFF	PHYCP_INTC	INTC	PHYCP System Interrupts Controller
0x6414_0000	0x6414_FFFF	AUDCP_INTC	INTC	AUDCP System Interrupts Controller
0x6416_0000	0x6416_FFFF	EFuse	EFuse	Electrical Fuse Controller
0x6417_0000	0x6417_FFFF	GPIO	GPIO	General Purpose IO Controller
0x6418_0000	0x6418_FFFF	AP_TMR0	TMR	AP System General Purpose Timer 0
0x6419_0000	0x6419_FFFF	AP_TMR1	TMR	AP System General Purpose Timer 1
0x641A_0000	0x641A_FFFF	AP_TMR2	TMR	AP System General Purpose Timer 2
0x641B_0000	0x641B_FFFF	KPD	KPD	Keypad Controller
0x641C_0000	0x641C_FFFF	AP_SYST	SYST	AP System System Timer
0x641E_0000	0x641E_FFFF	APCPU_WDG	WDG	APCPU System Watch Dog
0x641F_0000	0x641F_FFFF	IIS_MST	IIS_MST	IIS Master Controller
0x6420_0000	0x6420_FFFF	EIC0	EIC	External Interrupt Controller 0
0x6421_0000	0x6421_FFFF	EIC1	EIC	External Interrupt Controller 1
0x6422_0000	0x6422_FFFF	EIC2	EIC	External Interrupt Controller 2
0x6423_0000	0x6423_FFFF	EIC3	EIC	External Interrupt Controller 3
0x6424_0000	0x6424_FFFF	EIC4	EIC	External Interrupt Controller 4

Start Address	End Address	Module Name	IP Name	Description
0x6425_0000	0x6425_FFFF	EIC5	EIC	External Interrupt Controller 5
0x6426_0000	0x6426_FFFF	CPALL_EIC	EIC	All CP External Interrupt Controller
0x6427_0000	0x6427_FFFF	THM0	THM	Thermal Controller 0
0x6428_0000	0x6428_FFFF	THM1	THM	Thermal Controller 1
0x6429_0000	0x6429_FFFF	THM2	THM	Thermal Controller 2
0x642A_0000	0x642A_FFFF	THM3	THM	Thermal Controller 3
0x642B_0000	0x642B_FFFF	BUSMON_TMR0	BUSMON_TM R	Bus monitor Count Start Timer 0
0x642C_0000	0x642C_FFFF	BUSMON_TMR1	BUSMON_TM R	Bus monitor Count Start Timer 1
0x642D_0000	0x642D_FFFF	BUSMON_TMR2	BUSMON_TM R	Bus monitor Count Start Timer 2
0x642E_0000	0x642E_FFFF	Pin_Reg	Pin_Reg	Pin Register
0x6430_0000	0x6430_3FFF	Analog_Reg_G0L	Analog_Reg_G1 L	Analog Group 1 Left Register
0x6430_4000	0x6430_7FFF	Analog_Reg_G1	Analog_Reg_G2	Analog Group 2 Register
0x6430_8000	0x6430_BFFF	Analog_Reg_G1L	Analog_Reg_G2 L	Analog Group 2 Left Register
0x6430_C000	0x6430_FFFF	Analog_Reg_G2	Analog_Reg_G2	Analog Group 2 Register
0x6431_4000	0x6431_7FFF	Analog_Reg_G3	Analog_Reg_G3	Analog Group 3 Register
0x6431_8000	0x6431_BFFF	Analog_Reg_G4	Analog_Reg_G4	Analog Group 4 Register
0x6431_C000	0x6431_FFFF	Analog_Reg_G5	Analog_Reg_G5	Analog Group 5 Register
0x6432_0000	0x6432_3FFF	Analog_Reg_G5R	Analog_Reg_G6 R	Analog Group 6 Right Register
0x6432_4000	0x6432_7FFF	Analog_Reg_G5L	Analog_Reg_G6 L	Analog Group 6 Left Register
0x6432_C000	0x6432_FFFF	Analog_Reg_G8	Analog_Reg_G8	Analog Group 8 Register
0x6433_0000	0x6433_3FFF	Analog_Reg_G9	Analog_Reg_G9	Analog Group 9 Register
0x6433_4000	0x6433_7FFF	Analog_Reg_G10	Analog_Reg_G1 0	Analog Group 10 Register
0x6433_8000	0x6433_BFFF	Analog_Reg_G11	Analog_Reg_G1 1	Analog Group 11 Register
0x6434_0000	0x6434_3FFF	Analog_Reg_G13	Analog_Reg_G1 3	Analog Group 12 Register
0x6434_4000	0x6434_7FFF	Analog_Reg_G1 M	Analog_Reg_G2 M	Analog Group 2 Middle Register

Start Address	End Address	Module Name	IP Name	Description
0x6434_8000	0x6434_BFFF	Analog_Reg_G2L	Analog_Reg_G2L	Analog Group 2 Left Register
0x6434_C000	0x6434_FFFF	Analog_Reg_G4L	Analog_Reg_G4L	Analog Group 4 Left Register
0x6435_0000	0x6435_7FFF	DDC_CTRL	DDC_CTRL	DDC Controller
0x6435_8000	0x6435_87FF	DDC_Analog	DDC_Analog	DDC Analog Register
0x6438_0000	0x6438_3FFF	Analog_Reg_G12_rf	Analog_Reg_G12_rf	Analog Group 12 Register
0x6439_0000	0x6439_FFFF	Analog_Reg_G12_cr	Analog_Reg_G12_cr	Analog Group 12 UFS PHY CR Register
0x6440_0000	0x644F_FFFF	ADI_MST	ADI_MST	ADI Master Controller
0x6450_0000	0x6450_FFFF	ISE	ISE	Secure Element System
0x6451_0000	0x6451_FFFF	RFTI	RFTI	RF Transceiver Interface
0x6452_0000	0x6452_FFFF	Serdes_CTRL0	Serdes_CTRL	Serdes Controller 0
0x6453_0000	0x6453_FFFF	Serdes_CTRL1	Serdes_CTRL	Serdes Controller 1
0x6455_0000	0x6455_FFFF	Chip_Reset_CTRL	Chip_Reset_CTRL	Chip Reset Controller
0x6456_0000	0x6456_FFFF	APCPU_TS	APCPU_TS	APCPU Time Stamp
0x6457_0000	0x6457_FFFF	DJTAG	DJTAG	DJTAG Controller
0x6458_0000	0x6458_FFFF	APB_BUSMON	APB_BUSMON	APB Bus Monitor
0x6459_0000	0x6459_FFFF	AP_WDG	WDG	AP Watch Dog
0x645A_0000	0x645A_0FFF	AON_Matrix_AXI_Regu	AXI_Regu	AON Matrix AXI QoS Regulator
0x645B_0000	0x645B_FFFF	AP_INTC6	INTC	AP System Interrupts Controller 6
0x645C_0000	0x645C_FFFF	AP_INTC7	INTC	AP System Interrupts Controller 7
0x645D_0000	0x645D_FFFF	BB_SW_RFSPI_MST	BB_SW_RFSPI_MST	Base Band SoftWare RFSPI Master
0x645E_0000	0x645E_FFFF	AON_RTM	TRM	Register Trace Monitor
0x645F_0000	0x645F_FFFF	AON_SW_RFFE	AON_SW_RFFE	AON SoftWare RFFE Register
0x6460_0000	0x6463_FFFF	MailBox	MailBox	Mail Box
0x6464_0000	0x6464_FFFF	BUSMON_TMR3	BUSMON_TMR	Bus monitor Count Start Timer 3
0x6480_0000	0x6481_FFFF	SEC_TZPC	SEC_TZPC	Security TZPC
0x6482_0000	0x6482_FFFF	SEC_Reg	SEC_Reg	Security Register

Start Address	End Address	Module Name	IP Name	Description
0x6483_0000	0x6483_FFFF	SEC_TMR	TMR	Security Timer
0x6484_0000	0x6484_FFFF	SEC_RTC	SEC_RTC	Security RTC
0x6485_0000	0x6485_FFFF	SEC_WDG	WDG	Security Watch Dog
0x6486_0000	0x6486_FFFF	SEC_GPIO	SEC_GPIO	Security GPIO
0x6487_0000	0x6487_FFFF	SEC_EIC	EIC	Security EIC
0x6488_0000	0x6488_FFFF	SEC_CLK_32K_Det	SEC_CLK_32K_Det	Security Clock 32k Detect
0x6489_0000	0x6489_FFFF	SEC_DBG_Reg	SEC_DBG_Reg	Security Debug Register
0x6490_0000	0x6490_FFFF	AON_APB_Reg	AON_APB_Reg	AON APB Register
0x6491_0000	0x6491_FFFF	PMU_Reg	PMU_Reg	Power Manage Unit
0x6492_0000	0x6492_FFFF	AON_CLK	AON_CLK	AON Clock Register
0x6493_0000	0x6493_FFFF	AON_Pre_Div	AON_Pre_Div	AON Clock Pre-Divider Register
0x6494_0000	0x6494_FFFF	AON_DVFS	AON_DVFS	AON DVFS Register
0x6495_0000	0x6495_FFFF	APCPU_DVFS	APCPU_DVFS	AP CPU DVFS Register
0x64A0_0000	0x64A0_FFFF	USB2.0	USB2.0	USB2.0 Controller
0x64A1_0000	0x64A1_FFFF	SpinLock	SpinLock	SpinLock Register
0x6500_0000	0x6501_5FFF	AON_IRAM	IRAM	AON IRAM

5.2.2 IRAM Address Map

Size	Function	APCPU View	CH View	SP View	PSCP View	PHYCP View	AUDCP View
2K B	sml_power_manager	0x00000000~0x000007FF	0x65000000~0x650007FF	0x65000000~0x650007FF	N/A	N/A	N/A
6K B	reserved	0x65000800~0x65001FFF	0x65000800~0x65001FFF	0x65000800~0x65001FFF	N/A	N/A	N/A
5K B	DDR Retention	0x65002000~0x650033FF	0x65002000~0x650033FF	0x65002000~0x650033FF	N/A	N/A	N/A
4K B	PSCP CR8 boot	0x65003400~0x650043FF	0x65003400~0x650043FF	0x65003400~0x650043FF	0x00000000~0x00000FFF	N/A	N/A
4K B	PHYCP CR8 boot	0x65004400~0x650053FF	0x65004400~0x650053FF	0x65004400~0x650053FF	N/A	0x00000000~0x00000FFF	N/A
4K B	Audio DSP boot	0x65005400~0x650063FF	0x65005400~0x650063FF	0x65005400~0x650063FF	N/A	N/A	0x65005400~0x650063FF

Siz e	Function	APCPU View	CH View	SP View	PSCP View	PHYCP View	AUDCP View
4K B	Audio DSP share	0x65006400~ 0x650073FF	0x65006400~ 0x650073FF	0x65006400~ 0x650073FF	0x65006400~ 0x650073FF	0x65006400~ 0x650073FF	0x65006400~ 0x650073FF
8K B	SMSG	0x65007400~ 0x650093FF	0x65007400~ 0x650093FF	0x65007400~ 0x650093FF	0x65007400~ 0x650093FF	0x65007400~ 0x650093FF	0x65007400~ 0x650093FF
4K B	pub debugging	0x65009400~ 0x6500A3FF	0x65009400~ 0x6500A3FF	0x65009400~ 0x6500A3FF	0x65009400~ 0x6500A3FF	0x65009400~ 0x6500A3FF	0x65009400~ 0x6500A3FF
1K B	pub efuse repair	0x6500A400~ 0x6500A7FF	0x6500A400~ 0x6500A7FF	0x6500A400~ 0x6500A7FF	0x6500A400~ 0x6500A7FF	0x6500A400~ 0x6500A7FF	0x6500A400~ 0x6500A7FF
10 KB	prebit/rank training	0x6500A800~ 0x6500CFFF	0x6500A800~ 0x6500CFFF	0x6500A800~ 0x6500CFFF	0x6500A800~ 0x6500CFFF	0x6500A800~ 0x6500CFFF	0x6500A800~ 0x6500CFFF
8K B	SMSG_1	0x6500D000~ 0x6500EFFF	0x6500D000~ 0x6500EFFF	0x6500D000~ 0x6500EFFF	0x6500D000~ 0x6500EFFF	0x6500D000~ 0x6500EFFF	0x6500D000~ 0x6500EFFF
24 KB	reserved	0x6500F000~ 0x65014FFF	0x6500F000~ 0x65014FFF	0x6500F000~ 0x65014FFF	0x6500F000~ 0x65014FFF	0x6500F000~ 0x65014FFF	0x6500F000~ 0x65014FFF
4K B	Security FW back	0x65015000~ 0x65015FFF	0x65015000~ 0x65015FFF	0x65015000~ 0x65015FFF	N/A	N/A	N/A
384 KB	CH_SYS IRAM	0x65016000~ 0x65075FFF	0x00000000~ 0x0005FFFF	N/A	N/A	N/A	N/A
64 KB	SP_SYS IRAM	0x62800000~ 0x6280FFFF	0x62800000~ 0x6280FFFF	0x00000000~ 0x0000FFFF	N/A	N/A	N/A

5.3 Interrupt List

5.3.1 INTC Interrupt Source List

Interrupt Name	Type	APCPU GIC Interrupt Serial Num	INTC Name	INTC Interrupt Serial Num
int_req_acc_prot_pmu	High	255	AP_INTC_7	31
int_req_acc_prot_aon_apb_reg	High	254	AP_INTC_7	30
int_pub_dmc_mpu_vio	High	253	AP_INTC_7	29
int_pub_mem_fw	High	252	AP_INTC_7	28
int_pub_ptm	High	251	AP_INTC_7	27
int_pub_dfi_busmon	High	250	AP_INTC_7	26
int_pub_dfs_giveup	High	249	AP_INTC_7	25
int_pub_dfs_error	High	248	AP_INTC_7	24

Interrupt Name	Type	APCPU GIC Interrupt Serial Num	INTC Name	INTC Interrupt Serial Num
int_pub_dfs_vote_done	High	247	AP_INTC_7	23
int_pub_dfs_deny	High	246	AP_INTC_7	22
int_pub_dfs_exit	High	245	AP_INTC_7	21
int_pub_dfs_complete	High	244	AP_INTC_7	20
vdsp_pwaitmode	High	243	AP_INTC_7	19
vdsp_pfatalerror	High	242	AP_INTC_7	18
isp_int_req_ch1	High	241	AP_INTC_7	17
isp_int_req_ch0	High	240	AP_INTC_7	16
mm_sys_vdsp_stop	High	239	AP_INTC_7	15
isp_dec_int_req	High	238	AP_INTC_7	14
dep_int_req_ch0	High	237	AP_INTC_7	13
busmon_int_isp_blk	High	236	AP_INTC_7	12
busmon_int_jpg	High	235	AP_INTC_7	11
busmon_int_vdsp_blk	High	234	AP_INTC_7	10
csi_sw_fatal_error	High	233	AP_INTC_7	9
int_req_ana_int2	High	232	AP_INTC_7	8
int_req_ana_int1	High	231	AP_INTC_7	7
int_req_ana_int0	High	230	AP_INTC_7	6
int_req_mem_fw_aon	High	229	AP_INTC_7	5
int_req_slv_fw_aon	High	228	AP_INTC_7	4
int_req_top_dvfs	High	227	AP_INTC_7	3
int_req_ai_powervr_0	High	226	AP_INTC_7	2
int_req_ai_powervr_1	High	223	AP_INTC_6	31
int_req_ai_powervr_2	High	222	AP_INTC_6	30
int_req_ai_mem_fw	High	221	AP_INTC_6	29
int_req_slv_fw_ap	High	220	AP_INTC_6	28
int_req_dma	High	219	AP_INTC_6	27
int_sec_req_dma	High	218	AP_INTC_6	26
int_req_emmc	High	217	AP_INTC_6	25
int_req_i2c0	High	216	AP_INTC_6	24
int_req_i2c1	High	215	AP_INTC_6	23

Interrupt Name	Type	APCPU GIC Interrupt Serial Num	INTC Name	INTC Interrupt Serial Num
int_req_i2c2	High	214	AP_INTC_6	22
int_req_i2c3	High	213	AP_INTC_6	21
int_req_i2c4	High	212	AP_INTC_6	20
int_req_i2c5	High	211	AP_INTC_6	19
int_req_i2c6	High	210	AP_INTC_6	18
int_req_i2c7	High	209	AP_INTC_6	17
int_req_i2c8	High	208	AP_INTC_6	16
int_req_i2c9	High	207	AP_INTC_6	15
int_req_iis0	High	206	AP_INTC_6	14
int_req_iis1	High	205	AP_INTC_6	13
int_req_iis2	High	204	AP_INTC_6	12
int_req_sdio0	High	203	AP_INTC_6	11
int_req_sdio1	High	202	AP_INTC_6	10
int_req_sdio2	High	201	AP_INTC_6	9
int_req_ai_perf_busmon	High	200	AP_INTC_6	8
int_req_spi0	High	199	AP_INTC_6	7
int_req_spi1	High	198	AP_INTC_6	6
int_req_spi2	High	197	AP_INTC_6	5
int_req_ap_uart0	High	196	AP_INTC_6	4
int_req_ap_uart1	High	195	AP_INTC_6	3
nCTIIRQ[7]	High	194	AP_INTC_6	2
nCTIIRQ[6]	High	191	AP_INTC_5	31
nCTIIRQ[5]	High	190	AP_INTC_5	30
nCTIIRQ[4]	High	189	AP_INTC_5	29
nCTIIRQ[3]	High	188	AP_INTC_5	28
nCTIIRQ[2]	High	187	AP_INTC_5	27
nCTIIRQ[1]	High	186	AP_INTC_5	26
nCTIIRQ[0]	High	185	AP_INTC_5	25
nPMUIRQ[7]	High	184	AP_INTC_5	24
nPMUIRQ[6]	High	183	AP_INTC_5	23
nPMUIRQ[5]	High	182	AP_INTC_5	22

Interrupt Name	Type	APCPU GIC Interrupt Serial Num	INTC Name	INTC Interrupt Serial Num
nPMUIRQ[4]	High	181	AP_INTC_5	21
nPMUIRQ[3]	High	180	AP_INTC_5	20
nPMUIRQ[2]	High	179	AP_INTC_5	19
nPMUIRQ[1]	High	178	AP_INTC_5	18
nPMUIRQ[0]	High	177	AP_INTC_5	17
nCLUSTERPMUIRQ	High	176	AP_INTC_5	16
nFAULTIRQ1to8	High	175	AP_INTC_5	15
nFAULTIRQ[0]	High	174	AP_INTC_5	14
nERRIRQ1to8	High	173	AP_INTC_5	13
nERRIRQ[0]	High	172	AP_INTC_5	12
fault_int	High	171	AP_INTC_5	11
err_int	High	170	AP_INTC_5	10
pmu_int	High	169	AP_INTC_5	9
int_req_apcpu_cps	High	168	AP_INTC_5	8
int_req_ap_uart2	High	167	AP_INTC_5	7
int_req_ap_uart3	High	166	AP_INTC_5	6
int_req_ce_pub	High	165	AP_INTC_5	5
int_req_ce_sec	High	164	AP_INTC_5	4
int_req_busmon_ufshc	High	163	AP_INTC_5	3
int_req_ufs_fe_int	High	162	AP_INTC_5	2
int_req_ufs_SInterrupt	High	159	AP_INTC_4	31
int_req_dsi0[0]	High	158	AP_INTC_4	30
int_req_dsi0[1]	High	157	AP_INTC_4	29
int_req_dsi0_pll	High	156	AP_INTC_4	28
int_req_dsi1[0]	High	155	AP_INTC_4	27
int_req_dsi1[1]	High	154	AP_INTC_4	26
int_req_dsi1_pll	High	153	AP_INTC_4	25
int_req_dispc	High	152	AP_INTC_4	24
int_req_gsp0	High	151	AP_INTC_4	23
int_req_gsp1	High	150	AP_INTC_4	22
int_req_vpu_enc0	High	149	AP_INTC_4	21

Interrupt Name	Type	APCPU GIC Interrupt Serial Num	INTC Name	INTC Interrupt Serial Num
int_req_vpu_enc1	High	148	AP_INTC_4	20
int_req_vpu_dec	High	147	AP_INTC_4	19
int_req_gpu	High	146	AP_INTC_4	18
slv_fw_ipa_main_interrupt	High	145	AP_INTC_4	17
slv_fw_ipa_cfg_interrupt	High	144	AP_INTC_4	16
usb_interrupt[0]	High	142	AP_INTC_4	14
int_req_pamu3	High	141	AP_INTC_4	13
int_req_pamu3_comfifo	High	140	AP_INTC_4	12
pam_wifi_int_req_to_ap	High	139	AP_INTC_4	11
ipa_mAP_interrupt	High	138	AP_INTC_4	10
ipa_mAP0_interrupt	High	137	AP_INTC_4	9
ipa_mAP1_interrupt	High	136	AP_INTC_4	8
ipa_mAP2_interrupt	High	135	AP_INTC_4	7
ipa_mAP3_interrupt	High	134	AP_INTC_4	6
ipa_mAP4_interrupt	High	133	AP_INTC_4	5
ipa_mAP5_interrupt	High	132	AP_INTC_4	4
ipa_mAP6_interrupt	High	131	AP_INTC_4	3
ipa_mAP7_interrupt	High	130	AP_INTC_4	2
int_req_busmon_dbg_iram	High	127	AP_INTC_3	31
int_req_busmon_ipa_m0	High	126	AP_INTC_3	30
int_req_busmon_pam_wifi	High	125	AP_INTC_3	29
int_req_busmon_pam_u3	High	124	AP_INTC_3	28
dp_interrupt	High	123	AP_INTC_3	27
hdcp22_hpi_irq	High	122	AP_INTC_3	26
trng_irq	High	121	AP_INTC_3	25
int_req_dpu	High	120	AP_INTC_3	24
tca_int	High	119	AP_INTC_3	23
slv_fw_mm_interrupt	High	118	AP_INTC_3	22
int_fd	High	117	AP_INTC_3	21
int_idma_vau	High	116	AP_INTC_3	20
int_mstd_vau	High	115	AP_INTC_3	19

Interrupt Name	Type	APCPU GIC Interrupt Serial Num	INTC Name	INTC Interrupt Serial Num
vdma_int	High	114	AP_INTC_3	18
vdma_vau_int	High	113	AP_INTC_3	17
cpp_arm_int	High	112	AP_INTC_3	16
jpg_arm_int	High	111	AP_INTC_3	15
dcam2_arm_int	High	110	AP_INTC_3	14
dcam3_arm_int	High	109	AP_INTC_3	13
dcam0_arm_int	High	108	AP_INTC_3	12
dcam1_arm_int	High	107	AP_INTC_3	11
dcam0_3a_int	High	106	AP_INTC_3	10
dcam1_3a_int	High	105	AP_INTC_3	9
csi0_int_req0	High	104	AP_INTC_3	8
csi0_int_req1	High	103	AP_INTC_3	7
csi0_cal_done_int_req	High	102	AP_INTC_3	6
csi0_cal_failed_int_req	High	101	AP_INTC_3	5
csi1_int_req0	High	100	AP_INTC_3	4
csi1_int_req1	High	99	AP_INTC_3	3
csi1_cal_done_int_req	High	98	AP_INTC_3	2
csi1_cal_failed_int_req	High	95	AP_INTC_2	31
csi2_int_req0	High	94	AP_INTC_2	30
csi2_int_req1	High	93	AP_INTC_2	29
csi2_cal_done_int_req	High	92	AP_INTC_2	28
csi2_cal_failed_int_req	High	91	AP_INTC_2	27
csi3_int_req0	High	90	AP_INTC_2	26
csi3_int_req1	High	89	AP_INTC_2	25
csi3_cal_done_int_req	High	88	AP_INTC_2	24
csi3_cal_failed_int_req	High	87	AP_INTC_2	23
outbox_scr_irq_sig[0]	High	86	AP_INTC_2	22
inbox_scr_irq_sig[0]	High	85	AP_INTC_2	21
g3_cfg_aer_rc_err_int	High	84	AP_INTC_2	20
g3_cfg_aer_rc_err_msi	High	83	AP_INTC_2	19
g3_msi_ctrl_int	High	82	AP_INTC_2	18

Interrupt Name	Type	APCPU GIC Interrupt Serial Num	INTC Name	INTC Interrupt Serial Num
g3_rx_msg_int	High	81	AP_INTC_2	17
g3_err_int	High	80	AP_INTC_2	16
g3_cfg_pme_int	High	79	AP_INTC_2	15
g3_cfg_pme_msi	High	78	AP_INTC_2	14
g3_cfg_link_auto_bw_int	High	77	AP_INTC_2	13
g3_cfg_bw_mgt_int	High	76	AP_INTC_2	12
g3_rx_beacon_int	High	75	AP_INTC_2	11
g3_radm_inta	High	74	AP_INTC_2	10
g3_radm_intb	High	73	AP_INTC_2	9
g3_radm_intc	High	72	AP_INTC_2	8
g3_radm_intd	High	71	AP_INTC_2	7
int_req_pcie_pwr_up	High	70	AP_INTC_2	6
int_req_ddc_ctrl	High	69	AP_INTC_2	5
int_req_busmon_ddc_axi	High	68	AP_INTC_2	4
etr_axi_mon_int	High	67	AP_INTC_2	3
int_req_ipa_pwr_up	High	66	AP_INTC_2	2
int_req_busmon_tmc_axi	High	63	AP_INTC_1	31
int_req_ese_wdg_rst_flag	High	62	AP_INTC_1	30
eic_int_ap	High	61	AP_INTC_1	29
int_req_aud_wdg_rst_extd	High	60	AP_INTC_1	28
int_req_vbc_audply	High	59	AP_INTC_1	27
int_req_vbc_audrcd	High	58	AP_INTC_1	26
int_req_mcdt_ap	High	57	AP_INTC_1	25
int_req_dmaap	High	56	AP_INTC_1	24
dma_chn2_grp2_int	High	55	AP_INTC_1	23
dma_chn1_grp2_int	High	54	AP_INTC_1	22
dma_chn2_grp1_int	High	53	AP_INTC_1	21
dma_chn1_grp1_int	High	52	AP_INTC_1	20
int_req_tdm_tx	High	51	AP_INTC_1	19
int_req_tdm_rx	High	50	AP_INTC_1	18
int_ise_sys_to_ap_ree_maincpu	High	49	AP_INTC_1	17

Interrupt Name	Type	APCPU GIC Interrupt Serial Num	INTC Name	INTC Interrupt Serial Num
int_ise_sys_to_ap_tee_maincpu	High	48	AP_INTC_1	16
int_req_aon_tmr_tmr0	High	47	AP_INTC_1	15
int_req_aon_sys_frt	High	46	AP_INTC_1	14
int_req_mailbox_in0	High	45	AP_INTC_1	13
int_req_mailbox_out0	High	44	AP_INTC_1	12
int_req_mailbox_out6	High	43	AP_INTC_1	11
int_req_scc_int0	High	42	AP_INTC_1	10
int_req_scc_int1	High	41	AP_INTC_1	9
int_req_aon_gpio	High	40	AP_INTC_1	8
int_req_ap_tmr0_tmr0	High	39	AP_INTC_1	7
int_req_ap_tmr0_tmr1	High	38	AP_INTC_1	6
int_req_ap_tmr0_tmr2	High	37	AP_INTC_1	5
int_req_ap_tmr1_tmr0	High	36	AP_INTC_1	4
int_req_ap_tmr1_tmr1	High	35	AP_INTC_1	3
int_req_ap_tmr1_tmr2	High	34	AP_INTC_1	2
int_req_ap_tmr2_tmr0	High	31	AP_INTC_0	31
int_req_ap_tmr2_tmr1	High	30	AP_INTC_0	30
int_req_ap_tmr2_tmr2	High	29	AP_INTC_0	29
int_req_kpd	High	28	AP_INTC_0	28
int_req_ap_syst	High	27	AP_INTC_0	27
int_req_apcpu_wdg	High	26	AP_INTC_0	26
int_req_aon_iis	High	25	AP_INTC_0	25
int_req_aon_eic	High	24	AP_INTC_0	24
thm0_int	High	23	AP_INTC_0	23
thm1_int	High	22	AP_INTC_0	22
thm2_int	High	21	AP_INTC_0	21
thm3_int	High	20	AP_INTC_0	20
int_req_adi_mst	High	19	AP_INTC_0	19
int_req_ext_rstb_apcpu	High	18	AP_INTC_0	18
int_req_sec_cnt_top	High	17	AP_INTC_0	17
int_req_jtag	High	16	AP_INTC_0	16

Interrupt Name	Type	APCPU GIC Interrupt Serial Num	INTC Name	INTC Interrupt Serial Num
int_req_apb_busmon_aon	High	15	AP_INTC_0	15
int_req_ap_wdg	High	14	AP_INTC_0	14
int_req_sec_tmr	High	13	AP_INTC_0	13
int_req_sec_wdg	High	12	AP_INTC_0	12
int_req_sec_gpio	High	11	AP_INTC_0	11
int_req_sec_eic	High	10	AP_INTC_0	10
int_clk_32k_det	High	9	AP_INTC_0	9
int_req_aon_usb2	High	8	AP_INTC_0	8
int_req_busmon_aon_main_mtx_master	High	7	AP_INTC_0	7
int_req_pwr_down_all	High	6	AP_INTC_0	6
int_req_pwr_up_all	High	5	AP_INTC_0	5
int_req_pub_pwr_up	High	4	AP_INTC_0	4
int_req_ap_pwr_up	High	3	AP_INTC_0	3
int_req_apcpu_mode_st	High	2	AP_INTC_0	2

5.3.2 EIC Interrupt List

EIC Instance Name	Num	EIC Source	Description
EIC0	0	PAD: U0RXD	UART0 Interface Interrupt
	1	PAD: U0CTS	UART0 Interface Interrupt
	2	PAD: U1RXD	UART1 Interface Interrupt
	3	PAD: U2RXD	UART2 Interface Interrupt
	4	PAD: RFCTL15(G0)	UART3 Interface Interrupt
	5	PAD: U4RXD	UART4 Interface Interrupt
	6	PAD: U4CTS	UART4 Interface Interrupt
	7	PAD: U0RXD	UART0 Interface Interrupt
EIC1	0	PAD: U0CTS	UART0 Interface Interrupt
	1	PAD: U5RXD	UART5 Interface Interrupt
	2	PAD: U6RXD	UART6 Interface Interrupt
	3	PAD: U7RXD	UART7 Interface Interrupt

EIC Instance Name	Num	EIC Source	Description
	4	PAD:SD1_D3	PICE3_WAKEUP Interrupt
	5	Reserved	Reserved
	6	PAD:SD1_D2	PCIE3_CLKREQ Interrupt
	7	PCIE_SYS.PCIE_LINK_IN_L2_STATUS	PCIE_LINK_IN_L2_STATUS
EIC2	0	PAD: EXTINT0	EXTINT0
	1	PAD: EXTINT1	EXTINT1
	2	PAD: KEYIN0	EXTINT2
	3	PAD: KEYIN1	EXTINT3
	4	PAD: KEYIN2	EXTINT4
	5	PAD: SPI0_CSN	EXTINT5
	6	PAD: SPI0_DO	EXTINT6
	7	PAD: SPI0_DI	EXTINT7
EIC3	0	PAD: SPI0_CLK	EXTINT8
	1	PAD: EXTINT9	EXTINT9
	2	PAD: EXTINT10	EXTINT10
	3	PAD: KEYOUT0	EXTINT11
	4	PAD: KEYOUT1	EXTINT12
	5	PAD: IIS1DI	EXTINT13
	6	PAD: SCL9	EXTINT14
	7	PAD: SDA9	EXTINT15
EIC4	0	PAD: BAT_DET	EMMC_BUA_DET_AP
	1	PAD: BAT_DET	SDIO2_BUA_DET_AP
	2	PAD: BAT_DET	SDIO0_BUA_DET_AP
	3	PAD: TF_DET/SIM_DET0/SIM_DET1	SDIO0_TF_DET_AP/TF_DET
	4	PAD: TF_DET/SIM_DET0/SIM_DET1	SIM0_DET_PSCP/TF_DET
	5	PAD: TF_DET/SIM_DET0/SIM_DET1	SIM1_DET_PSCP/TF_DET
	6	PAD: BAT_DET	SIM0_BUA_DET_PSCP
	7	PAD: BAT_DET	SIM1_BUA_DET_PSCP
EIC5	0	Reserved	
	1	Reserved	
	2	IPA_SYS.USB31_WAKEUP	USB31_WAKEUP

EIC Instance Name	Num	EIC Source	Description
	3	USB20_SE0	
	4	Reserved	
	5	Reserved	
	6	Reserved	
	7	Reserved	
EIC_CPALL	0	PAD: U6RXD	PSCP Uart0
	1	PAD: U7RXD	PSCP Uart1
	2	PAD: BAT_DET	SDIO_BAT_DET_PSCP
	3	PAD: TF_DET/SIM_DET0/SIM_DET1	SIM0_DET_PSCP/TF_DET
	4	PAD: TF_DET/SIM_DET0/SIM_DET1	SIM1_DET_PSCP/TF_DET
	5	PAD: WIFI_COEXIT	WIFI_COEXIT
	6	PAD: BAT_DET	SIM1_BUA_DET_PSCP
	7	PAD: BAT_DET	SIM0_BUA_DET_PSCP
SEC_EIC	0	PAD:IIS1DI	SECURITY_GPIO0
	1	PAD:IIS1DO	SECURITY_GPIO1
	2	PAD:IIS1CLK	SECURITY_GPIO2
	3	PAD:IIS1LRCK	SECURITY_GPIO3
	4	PAD:U2TXD	SECURITY_GPIO4
	5	PAD:U2RXD	SECURITY_GPIO5
	6	PAD:AUD_DSP_RTCK	SECURITY_GPIO6
	7	PAD:SD2_D1	SECURITY_GPIO7

5.4 ADI Channel List

Channel Type	Event List	Description	Num
Event Channel	aapc_apr_dcm	Aapc apr dcm	0
	Reserved	Reserved	1
	Reserved	Reserved	2
	int_req_thm_adi	Interrupt request thermal adi	3
	overheat_shutdown_final	Overheat shut down final	4

Channel Type	Event List	Description	Num
	Reserved	Reserved	5
	pscp_sim0_power_off	Pscp sim0 power off	6
	pscp_sim1_power_off	Pscp sim1 power off	7
	sdio0_ldo_power_off	sdio0 ldo power off	8
	sdio0_ldo_power_off_delay	sdio0 ldo poweroff delay	9
	sdio0_emmc_io_power_off	sdio0 emmc io power off	10
	sdio2_ldo_power_off	sdio2 ldo power off	11
	sdio2_ldo_power_off_delay	sdio2 ldo power off delay	12
	sdio2_emmc_io_power_off	sdio2 emmc io power off	13
	emmc_ldo_power_off	Emmc ldo power off	14
	emmc_ldo_power_off_delay	Emmc ldo power off delay	15
	emmc_emmc_io_power_off	Emmc emmc_io power off	16
	Reserved	Reserved	17 ~ 23
Hardware TX Channel	scc_volt0_tune_req	Scc volt0 tune request	0
	scc_volt1_tune_req	Scc volt1 tune request	1
	dvfs_ddc_pub_tune_req	Dvfs ddc pub tune request	2
	dvfs_ddc_gpu_tune_req	Dvfs ddc gpu tune request	3
	dvfs_ddc_mm_tune_req	Dvfs ddc mm tune request	4
	dvfs_ddc_modem_tune_req	Dvfs ddc modem tune request	5
	dvfs_ddc_ai_tune_req	Dvfs ddc ai tune request	6
	dvfs_ddc_cpu0_tune_req	Dvfs ddc cpu0 tune request	7
	dvfs_ddc_sram1_tune_req	Dvfs ddc SRAM1 tune request	8
	dvfs_ddc_cpu1_tune_req	Dvfs ddc cpu1 tune request	9
	dvfs_ddc_sram2_tune_req	Dvfs ddc SRAM 2 tune request	10
	dvfs_ddc_cpu2_tune_req	Dvfs ddc cpu2 tune request	11

5.5 Thermal Sensor List

Thermal Controller Name	Sensor SN	Sensor Name and Location
THM3	7	N/A

Thermal Controller Name	Sensor SN	Sensor Name and Location
	6	N/A
	5	nr_thmsensor_1
	4	nr_thmsensor_0
	3	lte_thmsensor
	2	mm_thmsensor
	1	ai_thmsensor
	0	top_thmsensor
THM2	7	N/A
	6	apcpu_cluster_thmsensor
	5	gpu_thmsensor
	4	apcpu_core3_thmsensor
	3	apcpu_core2_thmsensor
	2	apcpu_core1_thmsensor
	1	apcpu_core0_thmsensor
	0	apcpu_lit_core_pll_thmsensor
THM1	7	N/A
	6	N/A
	5	N/A
	4	N/A
	3	apcpu_core6_thmsensor
	2	apcpu_core5_thmsensor
	1	apcpu_core4_thmsensor
	0	apcpu_mid_core_pll_thmsensor
THM0	7	N/A
	6	N/A
	5	N/A
	4	N/A
	3	N/A
	2	N/A
	1	apcpu_core7_thmsensor
	0	apcpu_big_core_pll_thmsensor

5.6 MailBox Channel List

Core Name	Channel Num	FIFO depth
APCPU	0	128
SP STAR MCU	1	32
CH STAR MCU	2	32
PSCP CR8	3	96
PHYCP CR8	4	32
AUD DSP	5	64
APCPU	6	32

5.7 Analog-Digital Interface

5.7.1 Overview

ADI, Analog Digital Interface, is used to connect SOC with external PMIC chip. It has a master design and also a slave to co-work. By ADI, the chip can control the PMIC chip and write command or read status from it.

In ADI r5p0, transmission protocol is updated and no more supports previous 10/12 bit address mode and three wire mode. For previous version, please refer to ADI r4p0 design specification.

Each ADI frame contains 44-bit data including 8 bits start bit, 15 bit address field, W/R control bit, and 16 bit data field and Parity bit and ACK/NACK bit all together.

On master side, SCK is one-direction pad while SD is bi-direct pad. When master perform transmit operation, SD is output, but when receive, SD will switch pad direction by inactive “oe” signal so as to receive serial data from slave.

On slave side, SCK is a input pad, while SD is bi-direct pad. When master perform transmit operation, SD is input, but when master receives, SD will switch pad direction by active “oe” signal so as to transmit serial data to master.

ADI SCK should be programmable by changing clock divider value in ADI control register.

5.7.2 Features

- Sync Two Serial Interface, SCK and SD (bi-direct).
- Due to noise in PMIC side, add Parity bit for error detection, and use ACK/NACK to determine whether master side to auto re-transmit or not.
- Auto re-transmit counter is configurable by software.
- Support debug mode (FSM hangs while NACK).
- Each frame contains 8 bits start sequence, 15 bits address, 1 bit R/W flag, 16 bits configure data and Parity bit and ACK/NACK bit, totally 44 bits.
- Programmable serial interface clock, according to the following formula:

$$F_{SCK} = F_{ADI\ master} / (T_{div} + 2)$$

if $T_{div}=0$, $F_{SCK} = F_{ADI\ master} / 2$ (13M by default)

- Execute WRITE and READ operation to PMIC chip through ADI.
- ASync FIFO (8 entries) adopted to accelerating the chip WRITE operation.
- The chip may once write at most 8 registers.
- Support MCU Write Buffer interrupt, indicating the FIFO full, empty, overrun status.
- For each MCU read/write operation, delay about 1 μ s (when clk_adi runs at 76.8MHz) should be tolerated. When clk_adi runs at 26M, the delay about 3 μ s.
- The interface timing is configurable, please refer to control registers gssi_ctrl0 & gssi_ctrl1.
- Offer several independent master write/read data channels for AP/CPs, to deal with data collision, also the number is configurable (MST_NUM is a parameter need to be configured, and also the corresponding PADDR width in SOC integration).
- Spinlock logic is provided in design. It should be obtain before a transaction and should be release after ending (alternative feature).

5.7.3 Function Description

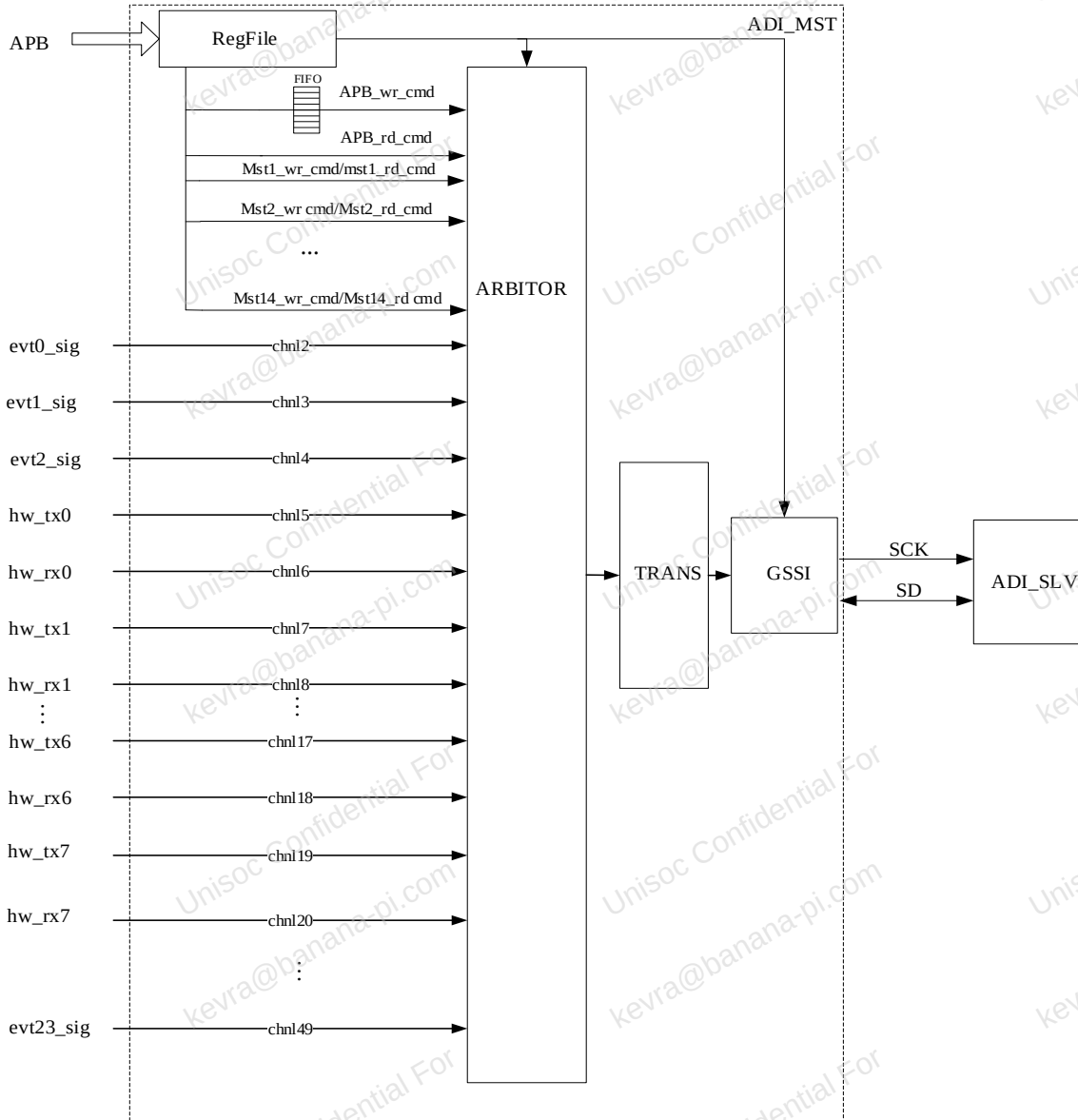
5.7.3.1 ADI Master

The main functions of ADI master are:

- Chip can access PMIC space through ADI
- The frame length including the address length, the data length and the w/r bit position is configurable (GSSI), the frame length is 44 bit.
- ARM_WR_CMD (Channel 0) is used to configure ADI and GSSI register, execute Write command
- ARM_RD_CMD (Channel 1) is used to execute Read command
- Hardware channels (Channel 2 ~ Channel 49) can be enabled or disabled independently
- Data collision between different AP/CPs also needs to be considered. Hence, there are up to 14 additional Master channels added, to support the write/read command to PMIC. One master can own a master channels. (Alternative feature, if not support master channels) To deal with spinlock while using ADI, here put the spinlock logic inside the design, which offers an additional REG address for read and write (no concern with ADI function).

5.7.3.2 ADI Block Diagram

Figure 5-1 ADI Block Diagram



5.7.3.3 Channels Description

- **ARM_WR_CMD** (master0): command mode / mapping mode
 - Command mode:
 - set **0x04 HW_MST_CTL cmd_wr_addr_en** to 1.
 - Configure **0x21C ARM_WR_CMD** then **0x220 ARM_WR_DATA**
 - Mapping mode:
 - **PADDR[20:17]=1**, **PADDR[16:2] -> 15 bit address**
 - **PWDATA[15:0] -> 16 bit data**

- **ARM_RD_CMD** (master0): Configure **0x28 ARM_RD_CMD**, then it triggers a APB read request. Data[15:0] is valid from **0x2C ARM_RD_DATA** after arm_rd_req is 0.
- **Master Read Command** channels (master1-14) are the same to ARM_RD_CMD channel. There is a read_empty flag declares the valid of data read (0 means the data is ready).
- **Master Write Command** channels: Mapping mode only
It load the value of PADDR[16:0] and PWDATA[15:0] directly.
Besides, write_finish(LOW active) is a flag for the master to detect whether the channel is busy.
Command write enable for different masters is as the follows.

PADDR[20:17]	Master channels	Description
<u>0001</u>	APB_WR_CMD	APB-WR_CMD command write enable
<u>0010</u>	MST1_WR_CMD	MST1_WR_CMD command write enable
<u>0011</u>	MST2_WR_CMD	MST2_WR_CMD command write enable
<u>0100</u>	MST3_WR_CMD	MST3_WR_CMD command write enable
<u>0101</u>	MST4_WR_CMD	MST4_WR_CMD command write enable
<u>0110</u>	MST5_WR_CMD	MST5_WR_CMD command write enable
<u>0111</u>	MST6_WR_CMD	MST6_WR_CMD command write enable
<u>1000</u>	MST7_WR_CMD	MST7_WR_CMD command write enable
<u>1001</u>	MST8_WR_CMD	MST8_WR_CMD command write enable
<u>1010</u>	MST9_WR_CMD	MST9_WR_CMD command write enable
<u>1011</u>	MST10_WR_CMD	MST10_WR_CMD command write enable
<u>1100</u>	MST11_WR_CMD	MST11_WR_CMD command write enable
<u>1101</u>	MST12_WR_CMD	MST12_WR_CMD command write enable
<u>1110</u>	MST13_WR_CMD	MST13_WR_CMD command write enable
<u>1111</u>	MST14_WR_CMD	MST14_WR_CMD command write enable

NOTE

In project, ADI PADDR width can be changed according to the master channels required, e.g. PADDR[19:0] for master 1-6, and PADDR[20] should tie 0.

- **EVT_SIG** channel
 - Number: up to 24
 - Signal: evt_sig
 - posedge : write data 1'h1 to register
 - negedge : write data 1'h0 to register

NOTE

EVT_SIG 0, 1, 3~13 have a FIFO of depth 4, no others.

- **HW_TX** channel
 - Number: up to 12

- Signal:
 - hw_tx_req // input
 - hw_tx_data[15:0] // input
 - hw_tx_ack // output
- **HW_RX channel**
 - Number: up to 12
 - Signal:
 - hw_rx_req // input
 - hw_rx_data[15:0] // output
 - hw_rx_ack // output

This table is the mapping relationship of hardware channels and corresponding channel numbers in the design of ADI.

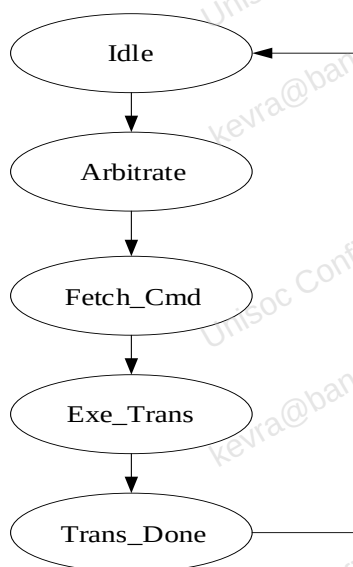
Hardware channel	Channel number
ARM_WR_CMD	chnl0
ARM_RD_CMD	chnl1
evt0_fifo	chnl2
evt1_fifo	chnl3
evt2_nofifo	chnl4
evt3_fifo	chnl29
evt4_fifo	chnl30
evt5_fifo	chnl31
evt6_fifo	chnl32
evt7_fifo	chnl33
evt8_fifo	chnl34
evt9_fifo	chnl35
evt10_fifo	chnl36
evt11_fifo	chnl37
evt12_fifo	chnl38
evt13_nofifo	chnl39
evt14_nofifo	chnl40
evt15_nofifo	chnl41
evt16_nofifo	chnl42
evt17_nofifo	chnl43
evt18_nofifo	chnl44
evt19_nofifo	chnl45

Hardware channel	Channel number
evt20_nofifo	chnl46
evt21_nofifo	chnl47
evt22_nofifo	chnl48
evt23_nofifo	chnl49
hw_tx0	chnl5
hw_tx1	chnl7
hw_tx2	chnl9
hw_tx3	chnl11
hw_tx4	chnl13
hw_tx5	chnl15
hw_tx6	chnl17
hw_tx7	chnl19
hw_tx8	chnl21
hw_tx9	chnl23
hw_tx10	chnl25
hw_tx11	chnl27
hw_rx0	chnl6
hw_rx1	chnl8
hw_rx2	chnl10
hw_rx3	chnl12
hw_rx4	chnl14
hw_rx5	chnl16
hw_rx6	chnl18
hw_rx7	chnl20
hw_rx8	chnl22
hw_rx9	chnl24
hw_rx10	chnl26
hw_rx11	chnl28
mst1_wr_chnl	chnl50
mst1_rd_chnl	chnl51
mst2_wr_chnl	chnl52

Hardware channel	Channel number
mst2_rd_chnl	chnl53
mst3_wr_chnl	chnl54
mst3_rd_chnl	chnl55
mst4_wr_chnl	chnl56
mst4_rd_chnl	chnl57
mst5_wr_chnl	chnl58
mst5_rd_chnl	chnl59
mst6_wr_chnl	chnl60
mst6_rd_chnl	chnl61
mst7_wr_chnl	chnl62
mst7_rd_chnl	chnl63
mst8_wr_chnl	chnl64
mst8_rd_chnl	chnl65
mst9_wr_chnl	chnl66
mst9_rd_chnl	chnl67
mst10_wr_chnl	chnl68
mst10_rd_chnl	chnl69
mst11_wr_chnl	chnl70
mst11_rd_chnl	chnl71
mst12_wr_chnl	chnl72
mst12_rd_chnl	chnl73
mst13_wr_chnl	chnl74
mst13_rd_chnl	chnl75
mst14_wr_chnl	chnl76
mst14_rd_chnl	chnl77

5.7.3.3.1 FSM

Figure 5-2 FSM State transmission diagram



The ADI master FSM is responsible for arbitration, fetch R/W command and serial transfer.

5.7.3.3.2 Arbitrator

The following commands will cause ADI master to transfer with analog die:

1. Arm® processor Write Command;
2. Arm® processor Read Command;
3. All other hardware channel WR/RD Commands;

If more than one of above commands is active simultaneously, arbitration is necessary.

This ADI Master has a 5-level configurable priority (Maximum is 4, Minimum is 0). If there is more than one active command in the same priority, command is selected by round-robin algorithm. The channel arbitration logic is implemented as follows.

1. Search the highest priority in all active commands.
2. Search the commands that have this priority level.
3. Select the last executed command number on this priority level.
There are four registers to store the last executed commands numbers on each priority levels.
4. Beginning with this last executed command number, search the next execute command number on this priority level.
5. This new searched command number is the next command.

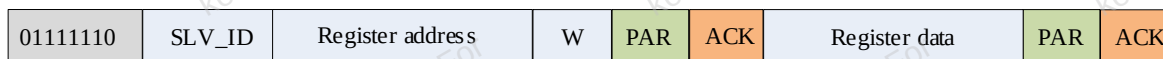
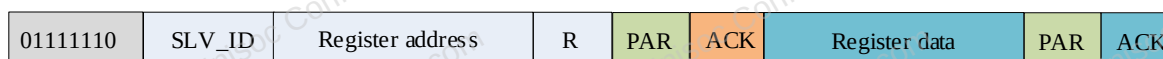
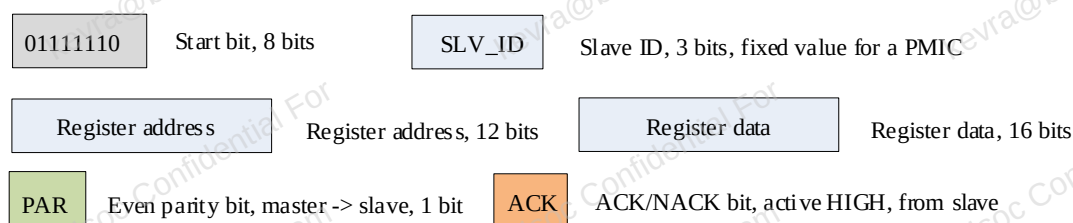
5.7.3.3.3 Transmission Protocol

All command and data send to analog side is stored in the Write command FIFO, it is an asynchronous FIFO with 8-depth and 32-bit width, used to accelerate ADI block write speed. MCU can continually send 8 write commands.

For register read, only one read data is buffered in ADI block, which means the chip can read one data at a time. After sending a read command, the chip should wait for a status bit change, indicating the serial interface data transfer done and read value is available. If the chip does not take this data before sending another read command, there will be a risk to lose the previous data.

SCK is programmable by changing of clock divider value in ADI control register.

The following graph shows how a write and read command is formed.



ADI master transmit data at the negedge of ADI_CLK, and receive data at posedge of ADI_CLK.

While ADI slave transmit data at the posedge of ADI_CLK and receive data at the posedge of ADI_CLK as well.

Figure 5-3 Write Operation

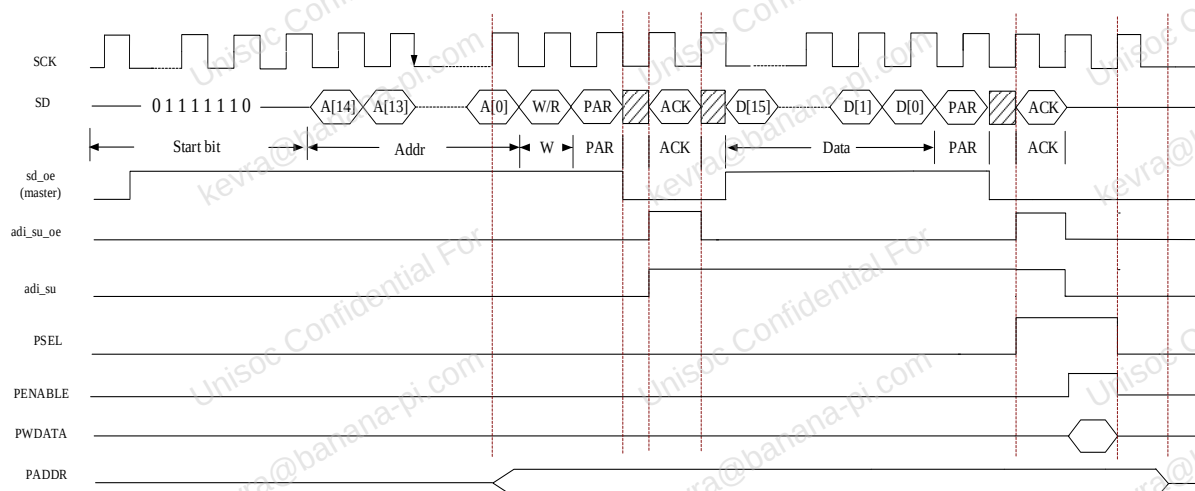
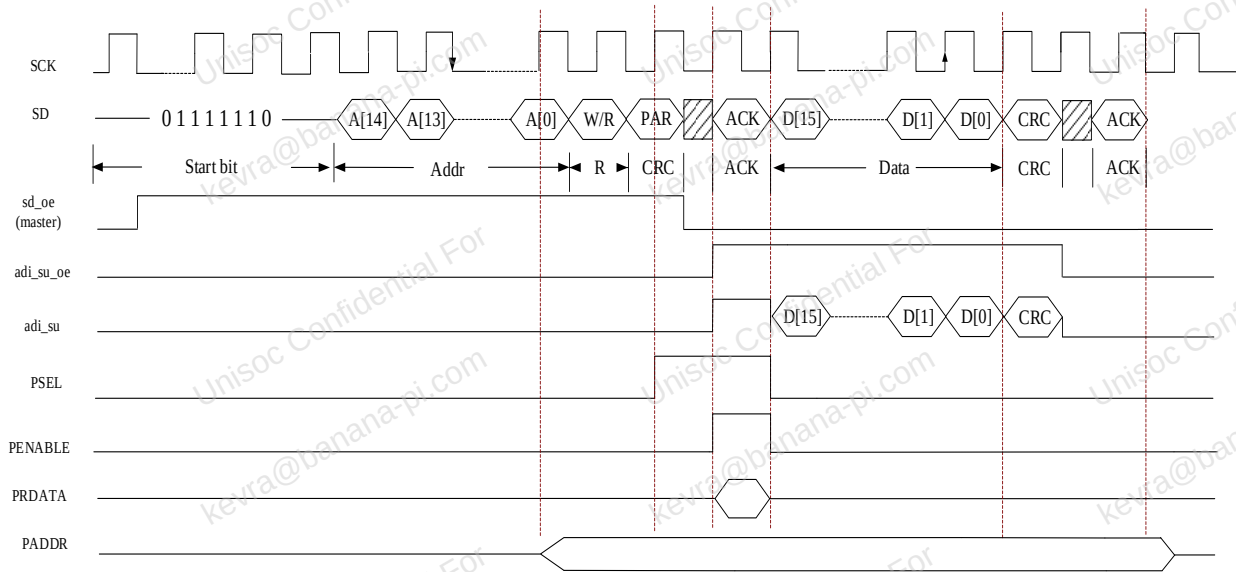


Figure 5-4 Read Operation



5.7.3.4 ADI Slave

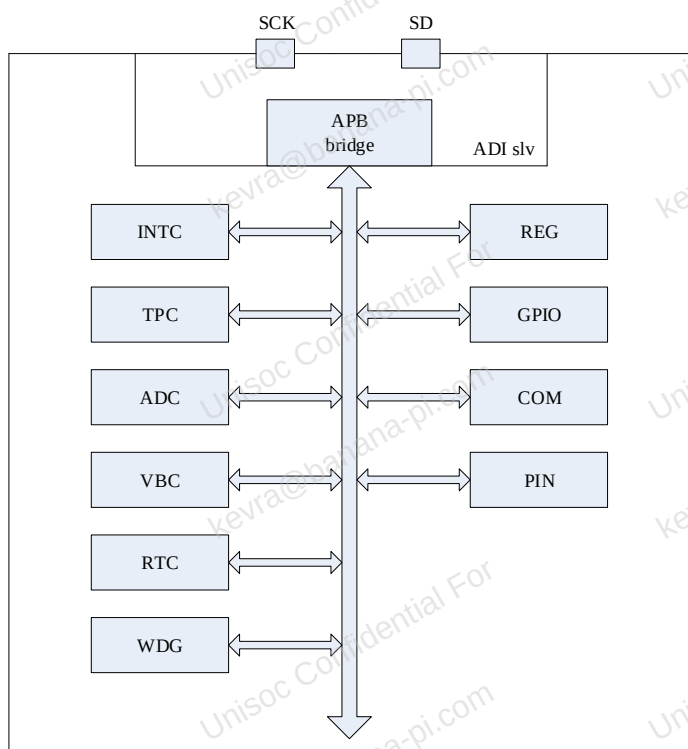
ADI slave function is to receive/transmit command and data from master; maintain control register and provide configure value to analog logic. The slave block is always droved by master.

The main function of ADI slave is:

1. Do serial to parallel shift when receive data from SD.
2. Do parallel to serial conversion when transmit data to SD.
3. Maintain parts of analog control registers.

The slave is quite simple, and main components are address shifter, transmit shifter, receive shifter and a transfer counter.

Figure 5-5 ADI Slave



5.7.3.5 Auto re-transmit mode

Auto re-transmission is a new feature, which enables ADI master to automatically re-transmit a transaction once NACK, and the maximum re-transmit counter is configurable (including 0).

When an ADI transaction is NACK, re-transmission automatically starts, again and again, until ACK or re-transmission counter reach the maximum value (set from 0x0 to 0x1f).

In details, for software channels, ARM_WR_CMD, ARM_RD_CMD and master channels, their request will be high for a long time, unless a re-transmission is ACK. If NACK still continually happens, and reach the maximum value, their request will hang. Software should clear this request manually, to make sure next transaction works normally.

For hardware channels, hw_tx and hw_rx, NACK continually happens will trigger a hw_tx_nack/hw_rx_nack, instead of hw_tx_ack/hw_rx_ack to previous design.

Besides, event channels lack a hand-shaking mechanism, hence software can observe NACK interrupt if necessarily.

NOTE

auto re-transmit mode and debug mode cannot be active at the same time.

In Auto re-transmit mode, software does not need to clear NACK and interrupt bit. Design will automatically recover from abnormal state to initial state.

5.7.3.6 Debug mode

Debug mode is convenient to observe design's status and configuration. Once NACK, design will hang until NACK is manually clear by software. During this time, FSM will enter ABNORMAL state and no more transactions can process. Once clearing the NACK, hw_tx/hw_rx can get the response of

hw_tx_nack/hw_rx_nack, and software channel's request also can be low. In addition, software channel's request also needs to be cleared in this mode.

5.7.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.7.5 Application Notes

ADI programming examples

Initialization steps

1. Configure **0x04 ADI_MST_CTL**, set
 - [5] -> 1, if using apb_cmd_wr (APB write command channel)
 - [4:3] -> 0, for 12 bit address mode (default)
 - [4:3] -> 1, for 10 bit address mode
 - [4:3] -> 2, for 15 bit address mode
2. Configure **0x20 GSSI_CTRL0**, set [30] -> 0, for PMIC clock gating.
3. Configure **0x24 GSSI_CTRL1**, set to 0x8003_0001 for two wire mode, set to 0x8003_0001 for two wire mode (use it only)

Channel setting

1. Know corresponding channel number for a specific hardware channel.
2. Configure **0x40 CHNL_EN** and **0x20C CHNL_EN1**.
Example, if **evt0_sig** (channel 2) is used in project, set **0x40 CHNL_EN** bit [2] -> 1, to enable the transfer enable.
3. Configure channel priority registers, if necessary.
0x08 ADI_MST_PRI_L, **0x0C ADI_MST_PRI_H**, **0x210 ADI_CHNL_PRI_2**,
0x214 ADI_CHNL_PRI_3, **0x218 ADI_CHNL_PRI_4**
4. Configure channel address (its corresponding PMIC register address)
Example: **hw0_tx** works (channel 5), then configure **0x50 CHNL5_ADDR**

Software channel mode program examples (APB_WR_CMD, APB_RD_CMD)

1. Configure **0x04 HW_MST_CTL**, according to cmd_wr_addr_en.
2. Configure APB command channels. For write, set **ARM_CMD_WR**'s chnl_addr0 and **AMR_DAT_WR**'s chnl_data if cmd_wr_addr_en is 1'h1 and using command mode. If cmd_wr_addr_en is 0, in the same cycle **PADDR[16:0]** and **PWDATA[15:0]** will be regarded as address and data that transmits on the bus and **PADDR[17]** should be 1 in 15 bit address mode.
3. For read, set **PADDR[8:0] = 0x28 ARM_RD_CMD**, then it triggers a APB read request. **Data[15:0]** can be read from **0x2C ARM_RD_DATA** after arm_rd_cmd_busy is 0.

Hardware Channels	Channel number	ADI register offset	PMIC address
evt0_sig	chnl2	0x44	321e_18cc
evt1_sig	chnl3	0x48	321e_18cc

Hardware Channels	Channel number	ADI register offset	PMIC address
evt3_sig	chnl29	0xb0	321e_1820
evt4_sig	chnl30	0xb4	321e_1820
evt5_sig	chnl31	0xb8	321e_197c
evt6_sig	chnl32	0xbc	321e_197c
evt7_sig	chnl33	0xc0	321e_1988
evt8_sig	chnl34	0xc4	321e_1994
evt9_sig	chnl35	0xc8	321e_19b8
evt10_sig	chnl36	0xcc	321e_19b8
evt11_sig	chnl37	0xd0	321e_19b8
evt12_sig	chnl38	0xd4	321e_19a0
evt13_sig	chnl39	0xd8	321e_19b8
evt14_sig	chnl40	0xdc	321e_19b8
evt15_sig	chnl41	0xe0	321e_19b8
evt16_sig	chnl42	0xe4	321e_1990
evt17_sig	chnl43	0xe8	321e_19b8
evt18_sig	chnl44	0xec	321e_19b8
hw_rx0	chnl6	0x54	321e_1844
hw_rx1	chnl8	0x5c	321e_1844
hw_tx0	chnl5	0x50	321e_1844
hw_tx1	chnl7	0x58	321e_1844
hw_tx4	chnl13	0x70	321e_1854
hw_tx5	chnl15	0x78	321e_1874
hw_tx6	chnl17	0x80	321e_1844
hw_tx7	chnl19	0x88	321e_1844

5.8 APB_BUSMON

5.8.1 Overview

Some applications are sensitive to latency, such as DSP. Once some transmission block APB, may lead to DSP abnormal software problem. APB Bus Monitor can serve as watchdog used for debug.

AXI Bus Monitor cannot monitor address and data at the same time. APB Bus Monitor can be used for monitoring address and data.

APB Bus Monitor will record latest APB transmission. Software can obtain address, data and other APB related information for debug.

5.8.2 Features

- Monitoring access timeout.
- Monitoring access address.
- Monitoring access data.

5.8.3 Function Description

5.8.3.1 Access timeout (apb_hang_det)

When a transmission is still not completed during preconfigured timeout threshold time, it will issue an interrupt and the address of current transmission will be saved for debug. The module will work only when apb_busmon_en and bus_hang_det_en are both valid.

When PREADY is pulled down, and PSEL is valid, bus_hang_det will work with the PCLK. When the value of counter reaches bus_hang_max_cycle (preconfigured value), the counter will stop working and bus_hang_int_req_raw will be pulled up.

Bus_hang_int_req_raw can be cleared by bus_hang_int_clr (INT_CLR[1]). Once apb_busmon_en (BUSMON_CFG[0]) or bus_hang_det_en (BUSMON_CFG[2]) isn't valid, bus_hang_raw_sts(INT_RAW[1]) will also be cleared.

APB_HANG_DET will record the following information when bus_hang_int_req_raw happens.

- paddr will be saved to addr_hang[31:0]
- prdata will be saved to data_hang[31:0], When APB read transmission
- pwdata will be saved to data_hang[31:0], when APB write transmission
- Other APB related information will be saved to bus_intf_hang[31:0]

bus_intf_hang									
position	31:(12+USER_WIDTH)	(11+USER_WIDTH):12	11	10	9	8:6	5:2	1	0
value	0	puser[USER_WIDTH-1:0]	psel	penable	pwrite	pprot[2:0]	pstrb[3:0]	pready	Pslverr

5.8.3.2 Monitor access address/data (apb_match_det)

The module will monitor accessing address/data. It will generate interrupt when masked PADDR/DATA is matching with preconfigured address/data or not. This module will work only when apb_busmon_en and bus_match_det_en are valid.

There will be extra bus_match_direction_sel[1:0] to decide which APB direction transmission to be monitored. Address_monitor_event and data_monitor_event will be triggered as follows:

- bus_match_direction_sel = 2'b00

Not monitor

- bus_match_direction_sel = 2'b01

Only monitor APB write transmission

- Address_monitor_event
 - addr_match_sel = 1
 $((paddr \& (\sim addr_mask)) == (addr_monitor \& (\sim addr_mask))) \& pwrite$
 - addr_match_sel = 0
 $((paddr \& (\sim addr_mask)) != (addr_monitor \& (\sim addr_mask))) \& pwrite$
- Data_monitor_event
 - data_match_sel = 1
 $((pwwdata \& (\sim data_mask)) == (data_monitor \& (\sim data_mask))) \& pwrite$
 - data_match_sel = 0
 $((pwwdata \& (\sim data_mask)) != (data_monitor \& (\sim data_mask))) \& pwrite$

- bus_match_direction_sel = 2'b10

Only monitor APB read transmission

- Address_monitor_event
 - addr_match_sel = 1
 $((paddr \& (\sim addr_mask)) == (addr_monitor \& (\sim addr_mask))) \& (\sim pwrite)$
 - addr_match_sel = 0
 $((paddr \& (\sim addr_mask)) != (addr_monitor \& (\sim addr_mask))) \& (\sim pwrite)$
- Data_monitor_event
 - data_match_sel = 1
 $((pwwdata \& (\sim data_mask)) == (data_monitor \& (\sim data_mask))) \& (\sim pwrite)$
 - data_match_sel = 0
 $((pwwdata \& (\sim data_mask)) != (data_monitor \& (\sim data_mask))) \& (\sim pwrite)$

- bus_match_direction_sel = 2'b11

Monitor APB write and read transmission both

apb_data = pwrite ? pwwdata: prdata;

- Address_monitor_event
 - addr_match_sel = 1
 $(paddr \& (\sim addr_mask)) == (addr_monitor \& (\sim addr_mask))$
 - addr_match_sel = 0
 $(paddr \& (\sim addr_mask)) != (addr_monitor \& (\sim addr_mask))$
- Data_monitor_event
 - data_match_sel = 1
 $(apb_data \& (\sim data_mask)) == (data_monitor \& (\sim data_mask))$
 - data_match_sel = 0
 $(apb_data \& (\sim data_mask)) != (data_monitor \& (\sim data_mask))$

The interrupt will be generated according to bus_match_type_sel[1:0]

Table 5-1 bus_match_int_req_raw interrupt generation

bus_match_type_sel	Interrupt condition
2'b00	address_monitor_event data_monitor_event
2'b01	data_monitor_event
2'b10	address_monitor_event
2'b11	address_monitor_event & data_monitor_event

Bus_match_int_req_raw can be cleared by bus_hang_int_clr (INT_CLR[0]) . Once apb_busmon_en (BUSMON_CFG[0]) or bus_match_det_en (BUSMON_CFG[1]) isn't valid, bus_match_int_req_raw will also be cleared.

APB_MATCH_DET will record the following information when bus_match_int_req_raw happens.

- paddr will be saved to addr_match[31:0]
- prdata will be saved to data_match[31:0], When APB read transmission
- pwdata will be saved to data_match[31:0], when APB write transmission
- Other APB related information will be saved to bus_intf_match[31:0]

5.8.3.3 Latest APB access (apb_latest_det)

The module will update APB transmission information when a new transmission begins. This module will work only when apb_busmon_en and bus_latest_det_en are valid.

APB_LATEST_DET will record the following information when new transmission happens.

- paddr will be saved to latest_addr[31:0]
- prdata will be saved to latest_data[31:0], When APB read transmission
- pwdata will be saved to latest_data[31:0], when APB write transmission
- Other APB related information will be saved to latest_bus_intf[31:0]

latest_bus_intf									
position	31:(12+USER_WIDTH)	(11+USER_WIDTH):12	11	10	9	8:6	5:2	1	0
value	0	puser[USER_WIDTH-1:0]	psel	penable	pwrite	pprot[2:0]	pstrb[3:0]	pready	pslverr

5.8.4 Control Registers

See UIS7885 AON System Control Registers Instructions.

5.8.5 Application Notes

5.8.5.1 debug bus

There will be sixteen 32 bits debug bus, which can be used for debug.

Table 5-2 Debug bus list

dbg_addr[3:0]	dbg_data[31:0]
0	addr_hang
1	data_hang
2	bus_intf_hang
3	addr_match
4	data_match
5	bus_intf_match
6	latest_addr
7	latest_data
8	latest_bus_intf
9	32'b0
...	32'b0
15	32'b0

5.8.5.2 apb_hang_det example

configure bus_hang_max_cycle to 4

BUSMON_CFG = 32'h0004_0005

INT_EN = 32'h0000_0002

When APB transmission is larger than 4 cycle, it will produce interrupt.

5.8.5.3 apb_match_det example

- only monitor APB write transmission, only monitor address, produce interrupt when masked address is matched with masked monitor_address.

- BUSMON_CFG = 32'h0000_0163
- ADDR_MONITOR = 32'h20b0_0000
- ADDR_MASK = 32'h0000_ffff
- INT_EN = 32'h0000_0001

When APB access address 0x20b0_0000 ~ 0x20b0_ffff, it will produce interrupt.

- monitor APB write/read transmission, only monitor data, produce interrupt when masked data is matched with masked monitor_data or masked address is matched with masked monitor_address.

- BUSMON_CFG = 32'h0000_03c3
- ADDR_MONITOR = 32'h20b0_0000
- DATA_MONITOR = 32'h0000_a5a5
- ADDR_MASK = 32'h0000_ffff
- DATA_MASK = 32'hffff_0000

– INT_EN = 32'h0000_0001

When APB access address 0x20b0_0000 ~ 0x20b0_ffff, or write data is 0x0000_a5a5 ~ 0xffff_a5a5, it will produce interrupt.

5.8.5.4 apb_latest_det example

BUSMON_CFG = 32'h0000_0009

5.9 EFuse

5.9.1 Overview

The electrical fuse is a type of non-volatile memory fabricated in standard CMOS logic process. This electrical fuse macro is widely used in chip ID, memory redundancy, security code, configuration setting, and feature selection, etc.

The efuse controller generates necessary timing and relative control interface to software.

This efuse controller is designed for Depth × 32 bits electrical fuses (depth is a parameter for the module, default is 128 for 4K efuse), support 7nm TSMC / UMC efuse.

This version of controller integrates two efuse controllers (normal and secure efuse) into one controller in order to reduce the efuse memory area and the version of efuse controller.

5.9.2 Features

- Support TSMC / UMC efuse product. Tpgm and strobe_low value should be defined before use the module (default: TSMC).
- Support APB4.0 for security consideration. An arbiter is used to insure the security access to the efuse index. Register Fire-Wall is built in to protect register file.
- Support double-bit implementation or not. The signal is made up of an input port from SOC and a register control together. If double-bit is enabled, there will be a shadow block for each efuse block to ensure the correct PGM action to efuse block.
- Build-in self-test logic(bist) can be triggered by ATE tester (only for normal part of efuse for security requirement).
- Support all_0_check to check all efuse initial status and meanwhile meet the security requirement.
- Support build-in programming protect logic. Block0~Block3 is used to store protect information for all block. A flag will assert when APB try to PGM the protect block.
- Support build-in programming auto check logic. When PGM process ended the controller will read back data and compare. A flag will assert when compare fail.
- For TSMC efuse product, support margin A_READ1 mode .
- Support power on read logic for bonding option and device life – cycle value of CE0.
- The efuse controller must be in the AON_SYS for power on read data available.

5.9.3 Control Registers

See UIS7885 AON System Control Registers Instructions.

5.10 DJTAG

5.10.1 Overview

This document introduces the DJTAG system. It describes the JTAG signals and the timing of JTAG.

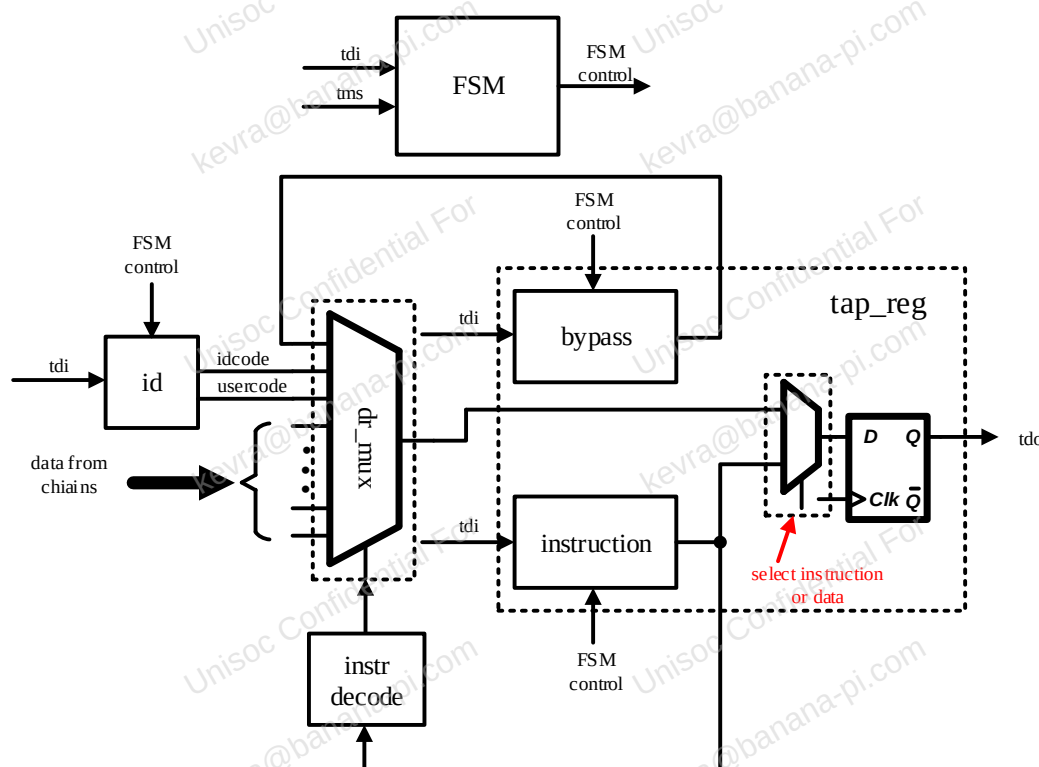
5.10.2 Features

- Up to 2^{CHAIN_NUM_WIDTH}-8 scan chains are supported in one TAP. The chain number is determined by a parameter MAX_CHAIN_NUM. The MAX_CHAIN_NUM maximum determined by a parameter CHAIN_NUM_WIDTH
- Support four kinds of function, bypass, idcode, usercode and chain scan.
- The scan chains can be updated or not according to the msb bit of instruction register by software configuration.

5.10.3 Function Description

The top design overall architecture is as follow.

Figure 5-6 DJATG architecture

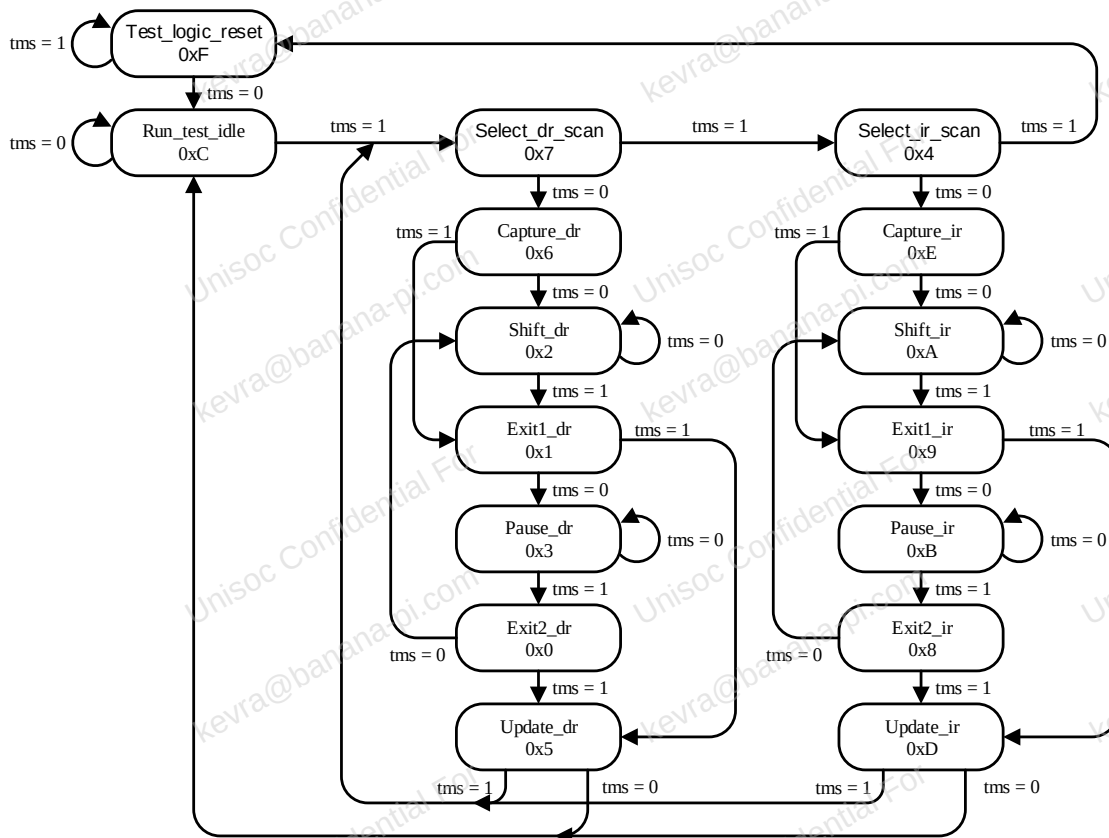


5.10.3.1 FSM

The TAP controller contains a FSM with 16 states. The figure below shows all the possible state changes of the FSM. The changes of state are controlled by **tms** signal.

For example, if the FSM is in the state of SELECT_DR_SCAN, at the rising edge of tck, the next state will be CAPTURE_DR if tms=0 or the next state will be SELECT_IR_SCAN if tms=1.

Figure 5-7 FSM



This FSM controls all the activities of TAP controller. It looks complicated but actually is quite simple. The two paths of FSM are similar except TEST_LOGIC_RESET and RUN_TEST_IDLE. In the figure, DR represents data registers and IR represents instruction registers.

If you want to get the output signal of one output port, you should first load the signal into the shift register on the boundary scan chain, then shift the signal out to get the signal you want.

If you want to load a signal to one input port, you should shift the signal into the certain register on the scan chain and load it into the input of system logic.

The details of these states as below.

TEST_LOGIC_RESET

In the TEST_LOGIC_RESET state the test logic is disabled to ensure the normal operation of on-chip system logic. The test logic can be reset by trstl signal. It can also be reset if the tms signal remains “1” for five clocks, so trstl is not a “must have” signal. In this state, at rising edge of tck, if tms remains high, the state will remain in the TEST_LOGIC_RESET state. If tms is low, the state will change to RUN_TEST_IDLE.

RUN_TEST_IDLE

This is a controller state between scan operations. It depends on the instruction if the test system do any activities. For example, the RUNBIST instruction causes an on-chip self-test in this state. Other instructions can also be designed to run activities in this state.

In this state, if tms is low at rising edge of tck, the state remains in RUN_TEST_IDLE, if tms is high at rising edge of tck, the state changes into SELECT_DR_SCAN.

SELECT_DR_SCAN

This is a temporary control state which all test data selected by the instruction remains their previous state.

If tms is low at rising edge of tck, the state is changed into CAPTURE_DR and a scan sequence of selected data registers is started. If tms is high at rising edge of tck, the state is changed into SELECT_IR_SCAN.

CAPTURE_DR

In this state, the data can be parallel-loaded into the data register selected by the instruction register on rising edge of tck. If the data register selected by the instruction doesn't have a parallel input, or the current instruction doesn't allow to capture data into data register, the data register remain its previous state.

In this state, when tck is applied a rising edge, the controller enters either SHIFT_DR state if tms is held low or EXIT1_DR if tms is held high.

SHIFT_DR

In this state, the data in the test data register selected by the current instruction between tdi and tdo is shifted to its serial output one stage at the rising edge of tck. Test data registers that are selected by the current instruction, but are not placed in the serial path, retain their previous state unchanged.

In this state, when tck is applied a rising edge, the controller enters either EXIT1_DR state if tms is held high or remains in the SHIFT_DR state if tms is held low.

EXIT1_DR

This is a temporary controller state. If tck is applied a rising edge, the state will either moves into UPDATE_DR state if tms is held high or PAUSE_DR state if tms is held low.

PAUSE_DR

In this controller state, the test data register selected by the instruction stopped shifting temporarily. All test data registers remain the previous state.

In this state, when tck is applied a rising edge, the state will either remains in PAUSE_DR state if tms is held low or move into EXIT2_DR state if tms is held high.

EXIT2_DR

This is a temporary state. When tck is applied a rising edge, the state will either moves to UPDATE_DR state if tms is held high or SHIFT_DR state if tms is held low.

UPDATE_DR

Some test data registers have latched parallel outputs to prevent changes of output when data is shifted in the certain shifting path. Data is latch onto the parallel output of the data registers in the UPDATE_DR state.

In this state, when tck is applied a rising edge, the state will either returns to RUN_TEST_IDLE state if tms is held low or SELECT_DR_SCAN state if tms is held high.

SELECT_IR_SCAN

This is a temporary state. The test registers selected by current instruction retain the previous state.

In this state, when tck is applied a rising edge, the state will either moves to CAPTURE_IR state if tms is held low or Test-Logic Reset state if tms is held high.

CAPTURE_IR

In this state, a fixed pattern of data will be loaded into the instruction register at rising edge of tck.

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the EXIT1_IR state if tms is held high or the SHIFT_IR state if tms is held low.

SHIFT_IR

In this state the instruction register is connected between tdi and tdo and shifts one data towards the serial output at the rising edge of tck.

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Exit1-IR state if tms is held high or remains in SHIFT_IR state if tms is held low.

EXIT1_IR

This is a temporary state. If a rising edge is applied to tck, the state will either moves into UPDATE_IR state if tms is held high or PAUSE_IR state if tms is held low.

PAUSE_IR

This state makes the instruction register to stop shifting temporarily.

In this state, if a rising edge is applied to tck, the state will either remains in this state if tms is held low or moves into EXIT2_IR state if tms is held high.

EXIT2_IR

This is a temporary controller state. In this state, if a rising edge is applied to tck, the state will either moves into UPDATE_IR if tms is held high or SHIFT_IR state if tms is held low.

UPDATE_IR

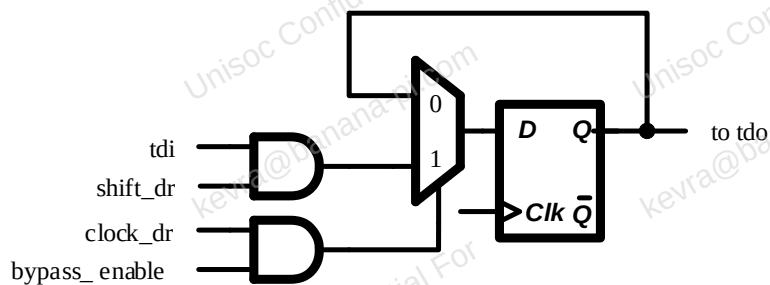
In this state, the data in the instruction register is latched onto the parallel output from the shift-register path on the falling edge of tck. Once an instruction is latched, it becomes the current instruction.

In this state, if a rising edge is applied to tck, the state will either moves into SELECT_DR_SCAN state if tms is held high or RUN_TEST_IDLE if tms is held low.

5.10.3.2 Bypass

The bypass register contains a signal shift-register stage and is used to provide a minimum-length serial path between the tdi and the tdo pins of a component when no test operation of that component is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test operations.

Figure 5-8 bypass circuit

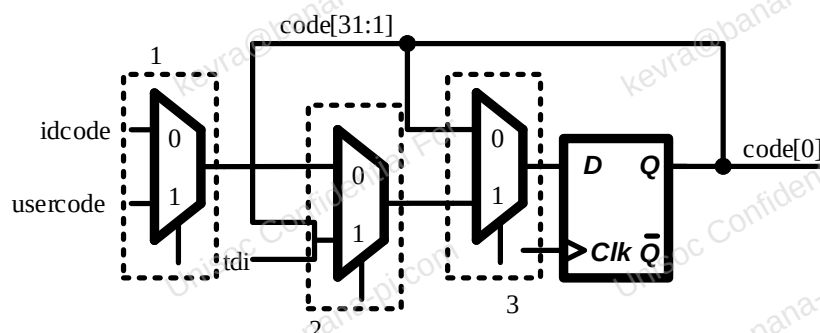


When JTAG select the bypass function, data from tdi will shift to tdo through the bypass register in SHIFT_DR state. The tdi is the data input port. When FSM switches to SHIFT_DR, the shift_dr signal is high. When FSM switches to shift_dr or capture_dr, the clock_dr signal is high. When the instruction is configured as bypass mode the bypass_enable is high.

5.10.3.3 Idcode and usercode

The idcode and usercode of the device are fixed 32 bit numbers. If the current instruction is idcode or usercode, the idcode or usercode data can be shifted out of tdo from the idcode register.

Figure 5-9 idcode/usercode circuit



The first mux select the idcode or usercode. The enable of first mux is usercode signal. The signal is high when instruction register select the usercode mode. The second mux is for data shift. When the FSM switches to SHIFT_DR, the select signal of second mux is high. When the Instruction register select the idcode or usercode, and the FSM switches to CAPTURE_DR or SHIFT_DR, the select signal of third mux is high.

5.10.3.4 Instruction register

The instruction can be shifted into instruction register. The instruction will choose the test data register to be accessed or the test to be performed.

The parallel output of the instruction register is latched to ensure the current instruction is not affected by the data shifting in the shift registers. The instruction register should at least have two register cells to meet the instruction need of the design.

The parallel output is updated at the end of the instruction scan cycle during the UPDATE_IR state. This action occurs at the falling edge of tck.

The instruction register allows instructions to be serially entered into the test logic during an instruction-register scan cycle. The instruction register is a (CHAIN_NUM_WIDTH + 1)-bits register, the MSB is update signal, the other 8 bits determine the function of DJTAG. 0x00: bypass, 0x01: idcode, 0x02: usercode, 0x03 – 0x07: reserve, other: select a scan chain.

Figure 5-10 instruction register

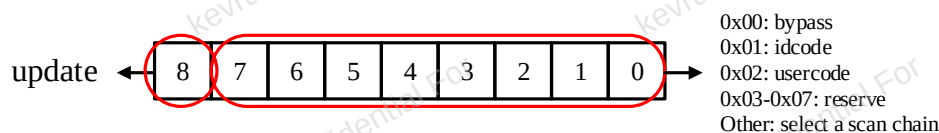
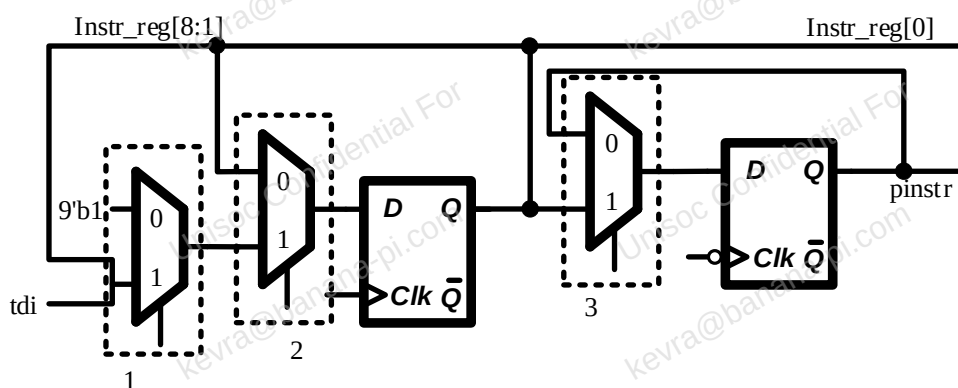


Figure 5-11 instruction circuit



The first mux select 9'b1 or data of instruction register after shift. The select signal is shiftir. When FSM switches to SHIFT_IR state, shiftir will be high. When the FSM switches to CAPTURE_IR or SHIFT_IR, the select signal of second mux is high. The third mux is for updating. If FSM switches to UPDATE_IR, system updates the instruction in the falling edge.

5.10.3.5 Instruction decode

The instruction register is a 9-bits register, the MSB is update signal, the other 8 bits determine the function of DJTAG. The instruction table is as follow:

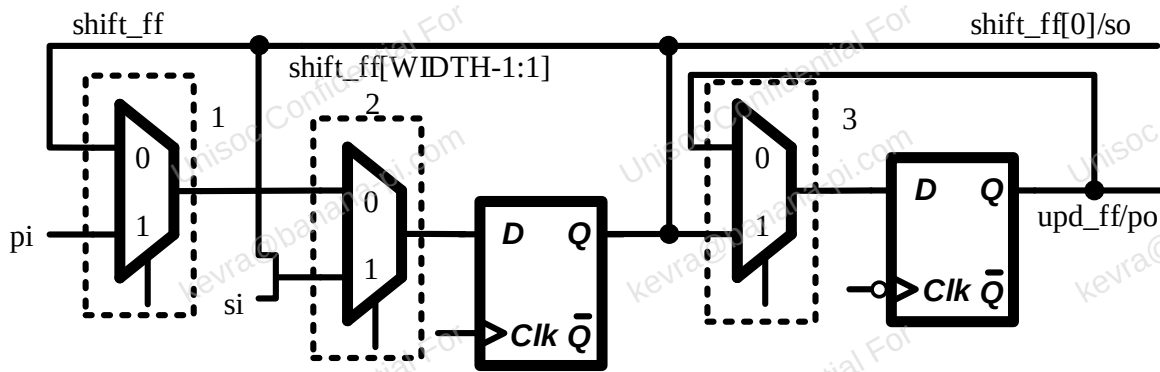
Table 5-3 instruction table

Instruction	Value	Description
bypass	(CHAIN_NUM_WIDTH + 1)'h00	bypass instruction
idcode	(CHAIN_NUM_WIDTH + 1)'h01	idcode instruction
usercode	(CHAIN_NUM_WIDTH + 1)'h02	usercode instruction
reserved	(CHAIN_NUM_WIDTH + 1)'h03 - (CHAIN_NUM_WIDTH + 1)'h07	reserved
boundary scan	others	chain scan

5.10.3.6 JTAG chain

The JTAG scan chains are test data registers which are put between input/output ports of logic and the on-chip logic. They allow system signals flow into and out of system logic without interfering the normal logic function. They are connected as a serial chain so that the data can be shifted in and out serially. The architecture of a scan chain register cell is displayed as follow.

Figure 5-12 scan chain circuit



A boundary scan register cell has four data input and output ports. They are pi, po, si, so. Pi is parallel input, po is parallel output, si is serial input and so is serial output.

When shiftdr is low and clockdr is high, which means the system is in CAPTURE_DR state, the shifting registers capture the signals on parallel inputs.

When shiftdr is high and clockdr is high, which means the system is in SHIFT_DR state, the shifting registers shift the data in shifting registers from si to so. When updatedr is high, which means the system is in UPDATE_DR state, and the instruction register's MSB is high, the data in shifting registers are loaded onto the parallel outputs of data register chain.

Figure 5-13 scan chain behavior

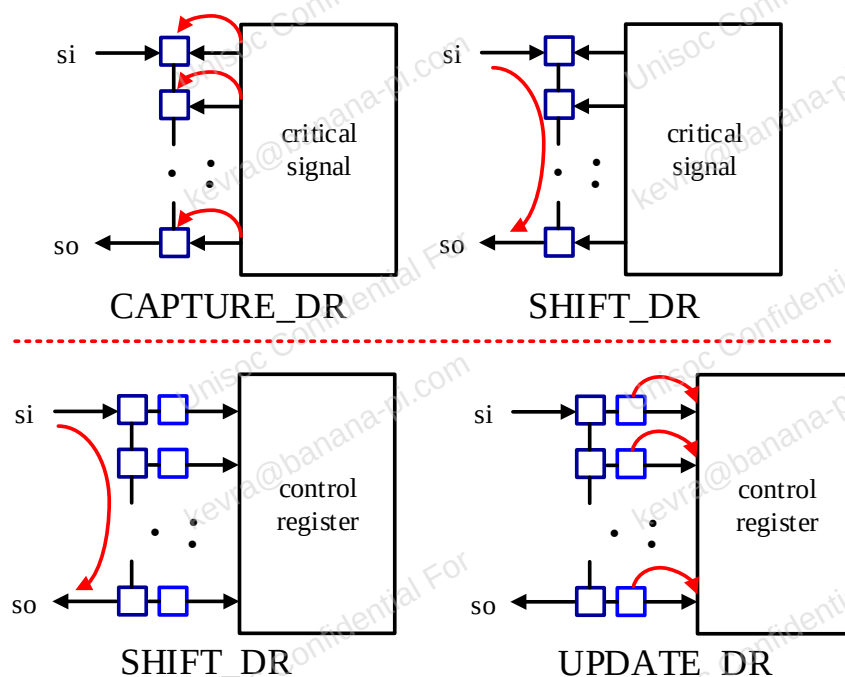


Table 5-4 Function table

Function	Instruction value	Descriptions
bypass	$(\text{CHAIN_NUM_WIDTH} + 1)'h00$	The bypass register contains a signal shift-register stage and is used to provide a minimum-length serial path between the tdi and the tdo pins of a component when no test operation of that component is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test operations.
idcode	$(\text{CHAIN_NUM_WIDTH} + 1)'h01$	The idcode of the device are fixed 32 bit numbers. If the current instruction is idcode, the idcode data can be shifted out of tdo from the idcode register.
usercode	$(\text{CHAIN_NUM_WIDTH} + 1)'h02$	The usercode of the device are fixed 32 bit numbers. If the current instruction is usercode, the usercode data can be shifted out of tdo from the usercode register.
reserve	$(\text{CHAIN_NUM_WIDTH} + 1)'h03 - (\text{CHAIN_NUM_WIDTH} + 1)'h07$	reserve
User defined chain	others	The JTAG scan chains are test data registers which are put between input/output ports of logic and the on-chip logic. They allow system signals flow into and out of system logic without interfering the normal logic function. They are connected as a serial chain so that the data can be shifted in and out serially.

5.10.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.10.5 Application Notes

Including module limitation, mode description, RAM input and output formats, special requirement, software control flow, description, etc.

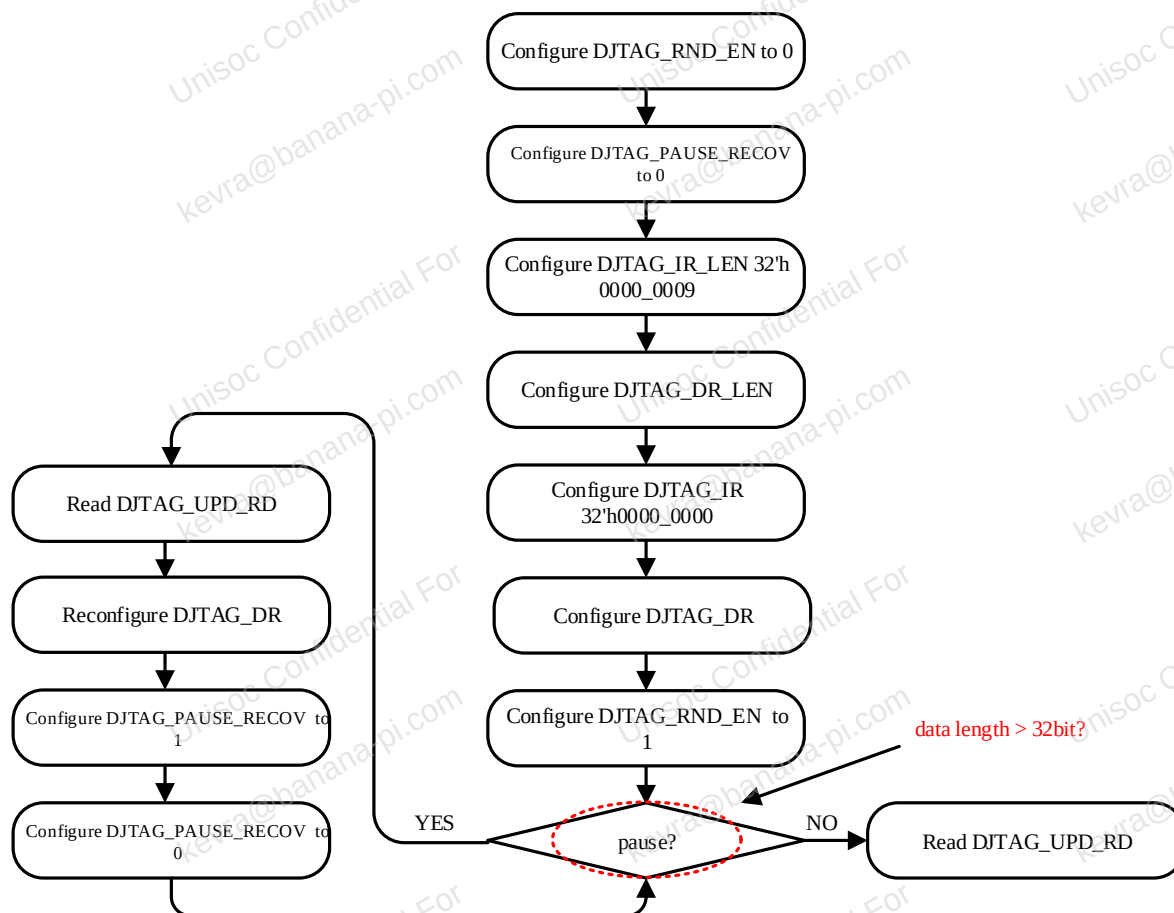
5.10.5.1 Bypass function

1. Configure DJTAG_RND_EN to 0 to disable the system.
2. Configure DJTAG_PAUSE_RECOV to 0 to keep the DJTAG in the PAUSE_DR state, if the data length is more than 32bits.
3. Configure DJTAG_IR_LEN to (CHAIN_NUM_WIDTH + 1).
4. Configure DJTAG_DR_LEN to you want.
5. Configure DJTAG_IR to 32'h0000_0000 to enter the bypass mode.
6. Configure DJTAG_DR to you want. If the data length is more than 32 bits, you should write from the 32 bits LSB.
7. Configure DJTAG_RND_EN to 1 to enable the DJTAG.
8. If the data length is more than 32 bits, read the DJTAG_UPD_DR for shifting data when the DJTAG switches to PAUSE_DR. Then you should write the DJTAG_PAUSE_RECOV to 1 after reconfigure DJTAG_DR. At next cycle or wait for a little time, you should write the DJTAG_PAUSE_RECOV to 0 to keep the DJTAG in the PAUSE_DR state in the next transition.

For example: if you want 128 bits data A[127:0], you should:

1. Write the A[31:0] to DJTAG_DR.
2. Keep the value until the DJTAG switches to PAUSE_DR.
3. Read the DJTAG_UPD_DR.
4. Reconfigure DJTAG_DR to A[63:32].
5. Configure the DJTAG_PAUSE_RECOV to 1.
6. At the next cycle or wait for a little time, configure the DJTAG_PAUSE_RECOV to 0.
7. Keep the whole registers' value until the DJTAG switches to PAUSE_DR state again. Then do as before repeatedly, but the DJTAG_DR should configure to A [95:64] in this transition and A [127:96] in next transition.
8. If the data length is less than 32 bits, usually, wait 80 cycles to read DJTAG_UPD_DR only.

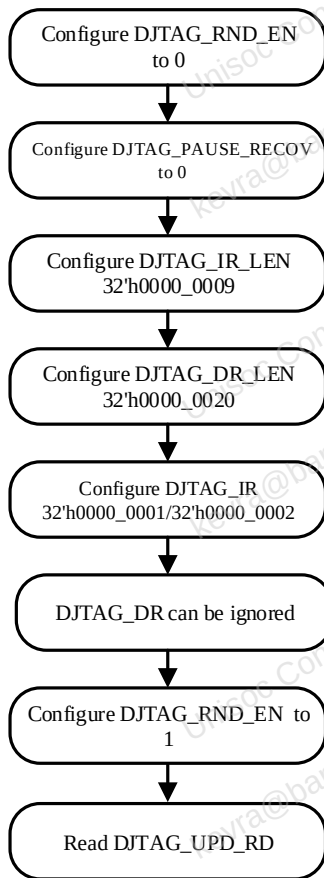
Figure 5-14 bypass mode control flow



5.10.5.2 Idcode/usercode function

1. Configure DJTAG_RND_EN to 0 to disable the system.
2. Configure DJTAG_PAUSE_RECOV to 0 to keep the DJTAG in the PAUSE_DR state, if the data length is more than 32bits.
3. Configure DJTAG_IR_LEN to (CHAIN_NUM_WIDTH + 1).
4. Configure DJTAG_DR_LEN to 32'h0000_0020.
5. Configure DJTAG_IR to 32'h0000_0001 for idcode function, or configure to 32'h0000_0002 for usercode function.
6. The value of DJTAG_DR can be ignored in these modes.
7. Configure DJTAG_RND_EN to 1 to enable the DJTAG.
8. Usually, wait 80 cycles to read DJTAG_UPD_DR after transition finish.

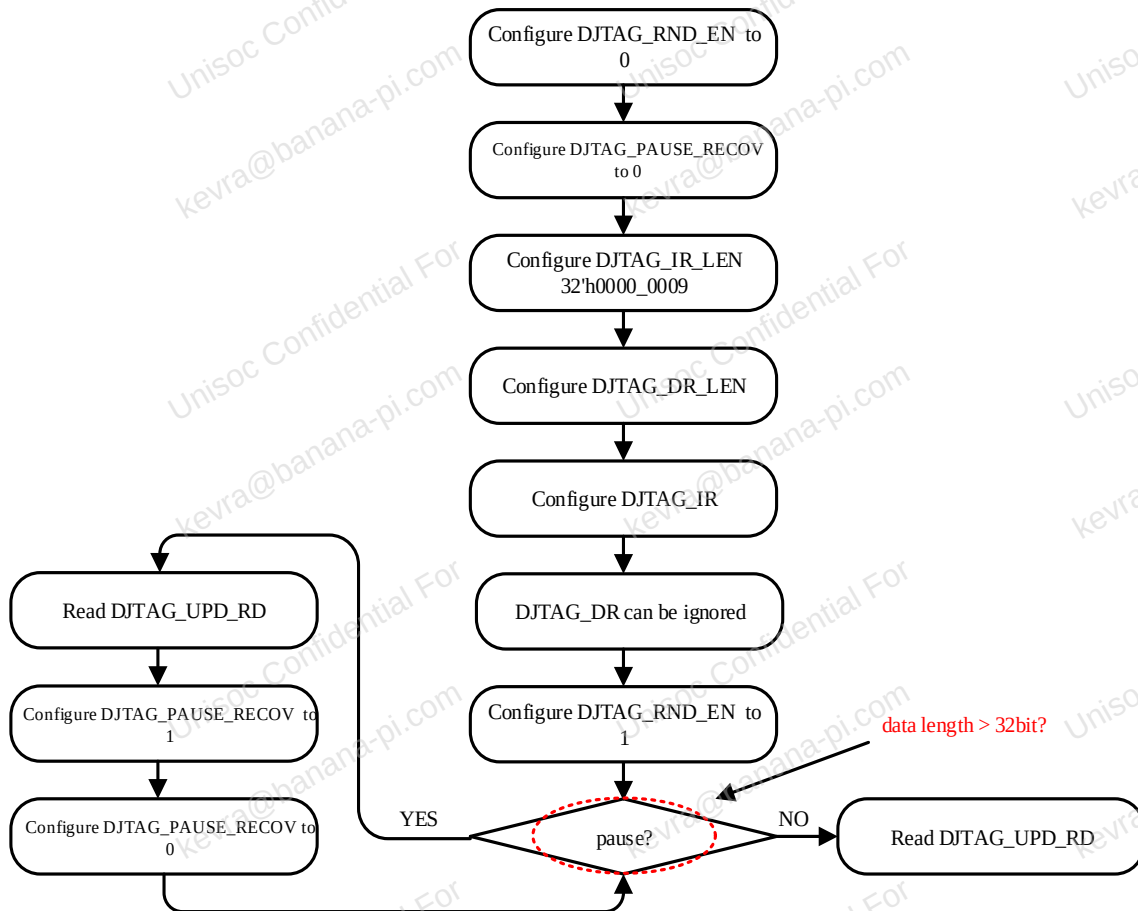
Figure 5-15 idcode/usercode mode control flow



5.10.5.3 Chain scan function (read)

1. Configure DJTAG_RND_EN to 0 to disable the system.
2. Configure DJTAG_PAUSE_RECOV to 0 to keep the DJTAG in the PAUSE_DR state, if the data length is more than 32bits.
3. Configure DJTAG_IR_LEN to(CHAIN_NUM_WIDTH + 1).
4. Configure DJTAG_DR_LEN to the value what is scan chain's length.
5. Configure DJTAG_IR to 32'h0000_0008~ (MAX_CHAIN_NUM + 8) (if the DJTAG has these chains) to select a scan chain and enter the chain scan (read) mode.
6. Because it's the read critical signal mode, DJTAG_DR can be ignored.
7. Configure DJTAG_RND_EN to 1 to enable the DJTAG.
8. If the data length is more than 32 bits, read the DJTAG_UPD_DR for shifting data when the DJTAG switches to PAUSE_DR. Then you should write the DJTAG_PAUSE_RECOV register to high. At next cycle or wait for a little time, you should write the DJTAG_PAUSE_RECOV to 0.
9. Usually, wait 80 cycles to read DJTAG_UPD_DR.

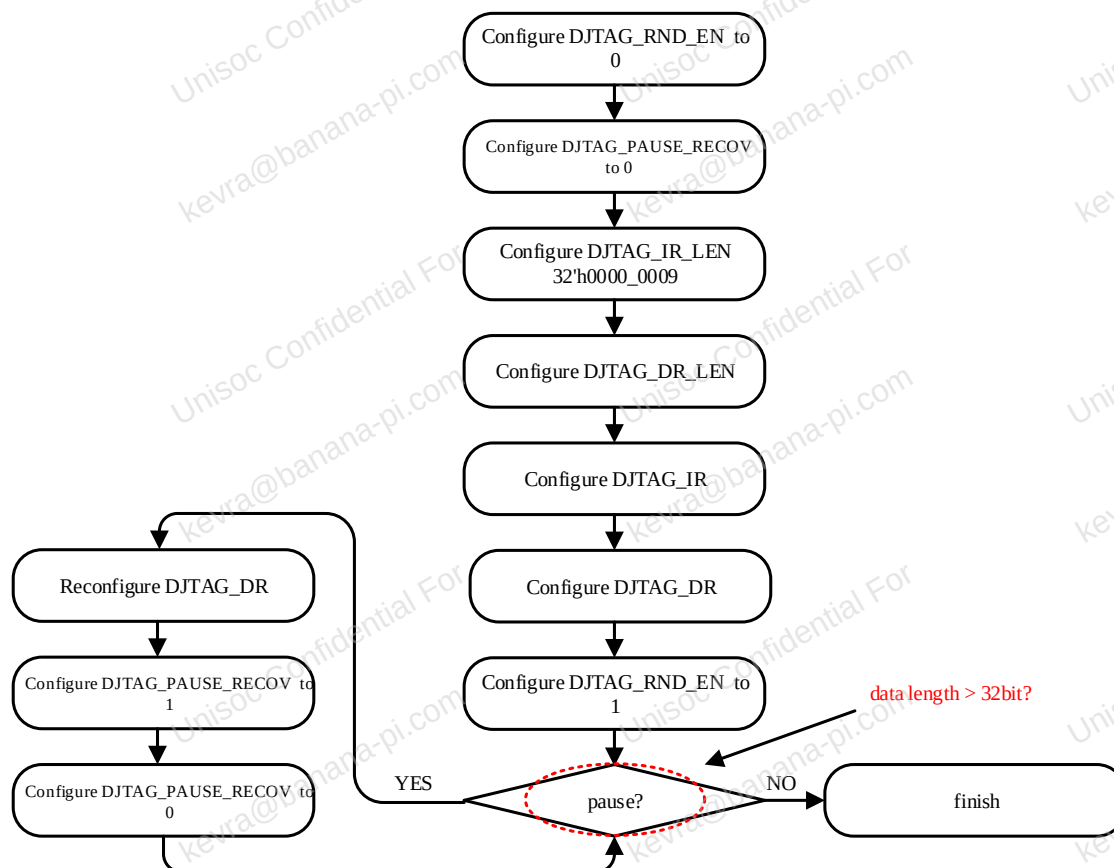
Figure 5-16 chain scan read mode control flow



5.10.5.4 Chain scan function (write)

1. Configure DJTAG_RND_EN to 0 to disable the system.
2. Configure DJTAG_PAUSE_RECOV to 0 to keep the DJTAG in the PAUSE_DR state, if the data length is more than 32bits.
3. Configure DJTAG_IR_LEN to (CHAIN_NUM_WIDTH + 1).
4. Configure DJTAG_DR_LEN to the value what is scan chain's length.
5. Configure DJTAG_IR to $(2^{\text{CHAIN_NUM_WIDTH}} + 8) \sim (2^{\text{CHAIN_NUM_WIDTH}} + \text{MAX_CHAIN_NUM} + 8)$ (if the DJTAG has these chains) to select a scan chain and enter the chain scan (write) mode.
6. Configure DJTAG_DR to you want. If the data length is more than 32 bits, you should write from the 32 bits LSB.
7. Configure DJTAG_RND_EN to 1 to enable the DJTAG.
8. If the data length is more than 32 bits. Then you should write the DJTAG_PAUSE_RECOV to 1 after reconfigure DJTAG_DR when the DJTAG switches to PAUSE_DR. At next cycle or wait for a little time, you should write the DJTAG_PAUSE_RECOV to 0.
9. DJTAG will update the chain data automatically.

Figure 5-17 chain scan write mode control flow



5.11 EIC

5.11.1 Overview

The EIC module includes four sub-modules: EIC_DBNC, EIC_LATCH, EIC_ASYNC and EIC_SYNC.

The EIC_DBNC sub-module provides up to 8 source input signal connection. A debounce mechanism is used to capture input signal's stable status(ms grade) and a single-trig mechanism is introduced into this sub-module to enhance the input event detect reliability. In addition, this sub-module's clock can be shut off automatically to reduce power dissipation. The debounce range is from 1 ms to 4 s with the step of 1 ms. The input signals shorter than 1 ms will be omitted at this sub-module.

THE EIC_LATCH sub-module is used to latch some special input signal and send interrupts to MCU core by channel 0, and it can provide up to 8 EIC_LATCH source input signal connection.

The EIC_ASYNC uses clk_32k clock to capture short signals(μs grade) to generate interrupt to MCU by level or edge trigger. It can wake up system for clk_32k always alive.

The EIC_SYNC is similar with GPIO's input function.

5.11.2 Features

5.11.2.1 EIC_DBNC

- High level detect and interrupt generate(not bypass mode)
- Low level detect and interrupt generate(not bypass mode)
- Interrupt mask(not bypass mode)
- Interrupt clear(not bypass mode)
- Level interrupt generate at system power down with once active trig(not bypass mode) (NOT advised)
- Keep interrupt stable when no interrupt clear(not bypass mode)
- Input data sample or mask(not bypass mode)
- Debounce number configurable(not bypass mode)
- Only once trig active(not bypass mode)
- Trig level condition configurable(not bypass mode)
- High level detect and interrupt generate(bypass mode)
- Low level detect and interrupt generate(bypass mode)
- Interrupt mask(bypass mode)
- Interrupt clear(bypass mode)
- Input data sample or mask(bypass mode)
- Level interrupt generate at system power down(bypass mode) (NOT advised)
- Force to open the clock of debounce

5.11.2.2 EIC_LATCH

- Special IRQ or FIQ interrupt on channel 0 is used to latch some input signal status and trigger IRQ or FIQ interrupt by level detecting (no edge detecting)
- Special IRQ or FIQ interrupt on channel 0 is multiplexed from 8 independent input sources
- Using Latch circuit to realize interrupt generation, INTMSK connected to the set port of Latch, and INTCLR connected to the clear port of Latch.
- Provide raw status for each of 8 input sources on special channel
- Provide polarity control bit for each of 8 input sources on special channel
- Provide only one clear bit to clear all latches on special channel
- Each input source can be enabled or disabled independently on special channel

5.11.2.3 EIC_ASYNC

- Positive edge detect and interrupt generate
- Negative edge detect and interrupt generate
- Both edges detect and interrupt generate
- High level detect and interrupt generate
- Low level detect and interrupt generate
- Interrupt mask
- Interrupt clear
- Input data sample or mask
- Level interrupt generate at system power down

- Edge interrupt generate with 32kHz clock at system power down
- Level detect conditions changeable
- Edge detect conditions changeable

5.11.2.4 EIC_SYNC

- Input signal synced with clk_apb
- Positive edge detect and interrupt generate
- Negative edge detect and interrupt generate
- Both edges detect and interrupt generate
- High level detect and interrupt generate
- Low level detect and interrupt generate
- Interrupt mask
- Interrupt clear
- Input data sample or mask
- Level interrupt generate at system power down
- Level detect conditions changeable
- Edge detect conditions changeable

5.11.3 Function Description

As the below diagram shows, SW communicates with EIC module by APB BUS. When the module has received active input signals, interrupts will be created to notify Arm® core.

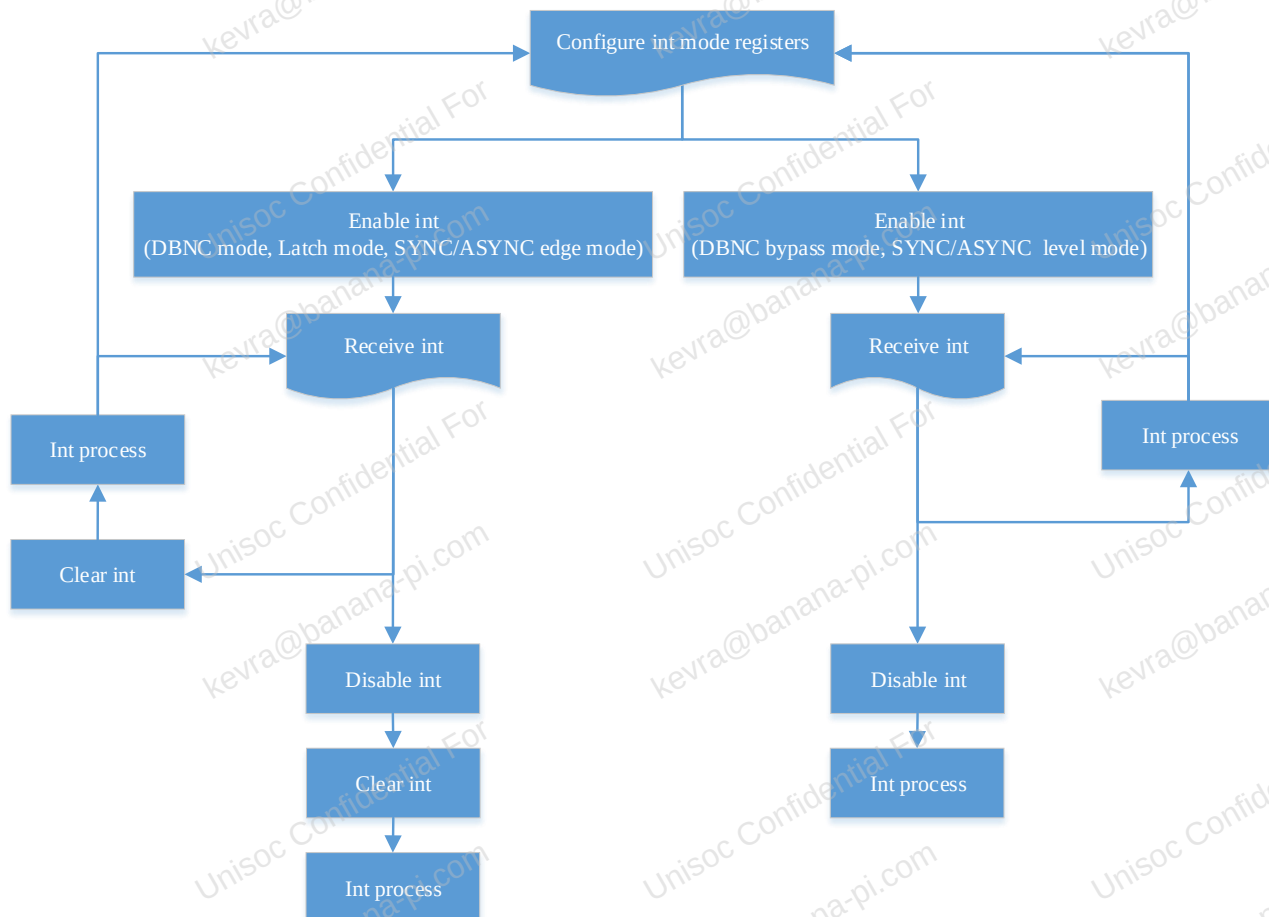
5.11.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.11.5 Application Notes

5.11.5.1 Software control flow

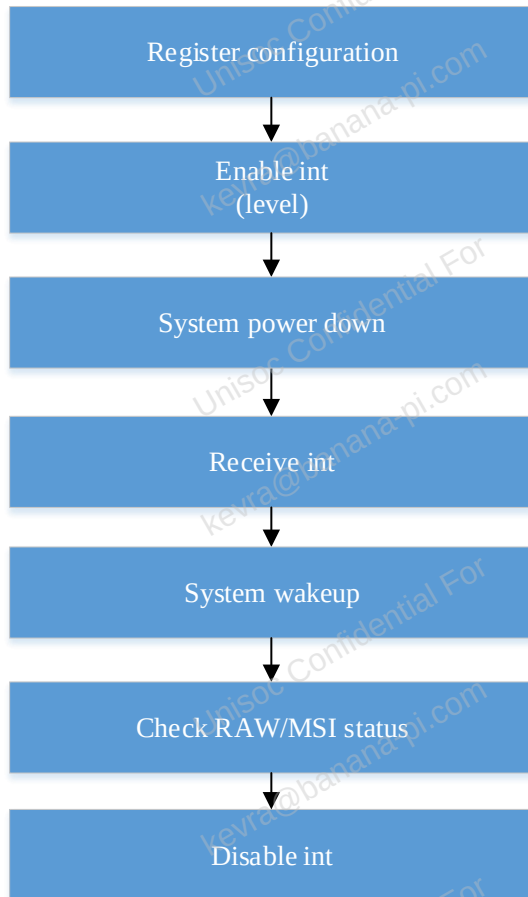
Figure 5-18 Software control flow



The diagram above shows a suggested flow of interrupt detection. Before detection, detection mode(level or edge), detection polarity, debounce number(for dbnc module), and other relative registers should be configured and one APB cycle may be taken to clear stored interrupt.

For level int detection of eic_dbnc clear operation is also needed. But for level int detection of eic_async and eic_sync, due to the combinational path of level detect, clear operation is only for edge int. If it's necessary int disable is an alternative operation for level detection clear/mask in these modules.

Figure 5-19 Software control flow(system wakeup)



The diagram above shows a suggested flow of system wakeup. Before power down (gate APB bus), detection polarity, and other relative registers should be configured and one APB cycle may be taken to clear stored interrupt (only for eic_latch module).

Especially for EIC_DBNC sub-module, software needs to adopt corresponding steps as follows:

1. When system resets, EIC_DBNC module is also under the reset status and cannot capture the EIC_DBNC input signals. At the process of system initialization, Arm® processor needs to configure the EIC_DBNC input signal detecting conditions, such as '1'/'0' detection, read INT status registers and write EIC_DBNC_IC to clear the EIC_DBNC INTs. Then, if necessary, Arm® processor sends a trig_start pulse to commence one EIC signal detection process. Before system enters deep-sleep mode or closes PCLK, it ought to assure that EIC_DBNC INT has been enabled and cleared and system has sent out active trig_start pulse.
2. After receiving the trig_start pulse, EIC_DBNC module starts the process of EIC_DBNC signal detection. If EIC captures one stable input signal, it will send an INT to Arm® processor, shut off the clock of debounce circuit automatically and wait for next trig_start pulse.
3. When Arm® processor receives the EIC_DBNC INT, it enters INT process flow. And if EIC input condition changes, Arm® processor needs to configure those detecting condition registers again.
4. If necessary, Arm® processor sends a trig_start pulse again to commence a new EIC_DBNC signal detection process, step 1 to step 3 cycles.
5. For the debounce bypass mode, Arm® processor can receive INT without the need of sending trig_start pulse.

5.11.5.2 Programming Notes

- For EIC_DBNC:
The register, EIC_DBNC_IEV, is used to set the input signal constraints for INT trigger.
EIC_DBNC_INT comes from EIC_DBNC_MIS, which EIC_DBNC_MIS is equal to EIC_DBNC_IE & EIC_DBNC_RIS. So, if it needs to get some EIC_DBNC' INT, the EIC_DBNC_IE should be unmasked.
And if Arm® processor has received one EIC_DBNC INT, it maybe need to mask the corresponding bit of EIC_DBNC_IE to avoid to receive the same INT, and set corresponding bits 1 to of EIC_DBNC_IC.
To quit EIC_DBNC FSM correctly, it needs several additional milliseconds. So it needs consider the additional time for exact debounce period.
- For EIC_LATCH:
The interrupt bit could tie to 1 unless a clear operation is performed.
Narrow pulse interrupt is also adapted with this module.
Suitable for system wakeup operation.
- For level detection of EIC_ASYNC and EIC_SYNC:
Level interrupt MUST be held after system check the RAW/MSI registers.

5.12 GPIO

5.12.1 Overview

The GPIO module provides general purpose input and output signals. Many of the GPIO pins are multiplexed with other functions, and system design trade-off must be exercised on selecting them. All the GPIO pins can be programmed to be either input or output. When in input mode, they can be programmed to trigger interrupt to the Arm® processor.

5.12.2 Features

- Configure detect condition by corresponding register
- Positive edge detect and interrupt generate
- Negative edge detect and interrupt generate
- Both edge detect and interrupt generate
- High level detect and interrupt generate
- Low level detect and interrupt generate
- Level detect and interrupt generate at system power-down
- Mask interrupt
- Mask input data
- Mask output data (mask to '0')

5.12.3 Function Description

Arm® core configures the function of GPIO by writing the registers through APB interface. With corresponding configuration, this module can perform data input and output as IO interface, detect the level or edge interrupt signal. If it is necessary, it wakes system up with a specified level input.

5.12.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.12.5 Application Notes

Programming Notes

1. In edge detection mode, clear operation should be performed before every interrupt detection cycle to prevent unexpected interrupt trigger.
2. During one interrupt detection process, any change of configuration registers are not recommended, that may course unexpected error.
3. Configuring interrupt mode:
 - a If level mode, set gpio_is to 1; If edge mode, set gpio_is to 0.
 - b If high level(posedge), set gpio_iev to 1; If low level(negedge), set gpio_iev to 0
 - c If both-edge mode, set gpio_ibe to 1, it will override gpio_iev.
 - d Clear operation is needed necessary for edge mode, it is supposed to write gpio_ic to 1, and rewrite it to 0 for next detection.
4. Level interrupt could not be clear by gpio_ic. To disable/clear level interrupt, only gpio_ie is suggested.
5. Interrupt enable signal is used to mask all interrupts defined in this module. A common usage of that is blocking interrupt and preventing interrupt transmitted to system.
6. Level interrupt signal must be held until Check Raw/Masked Int Registers process done. If the input becomes inactive, the interrupt source will disappear immediately.
7. Especially, in system wakeup situation, relative registers must be configured before system power down. Interrupt signal must be held until APB clock stable and Check Raw/Masked Int Registers done.

5.13 IIS_MST

5.13.1 Overview

IIS is a common digital audio interface specification. Base Band Chip IIS Interface can be used to implement a CODEC interface with external digital audio system working as host or slave. The IIS master interface supports IIS data format. The interface can transmit and receive data simultaneously as well as transmit or receive data alternatively at a time. This document defines function and configuration of IIS block. IIS module can be controlled both by Arm[®] processor or DSP.

5.13.2 Features

- Compliant with the IIS standard
- Support IIS, MSB-Justified and DSP data bus interface working on IIS mode
- Serial clock with programmable frequency
- Two 128-word-deep FIFOs minimizes processor overhead at high data rates
- LRCK polarity can be configured
- Wide selection of IIS data lengths 8,16, 32bits
- Support Master only
- Support MSB / LSB mode
- Transmit only / receive only / transmit and receive mode

- NG_TX, NG_RX can be configured
- 2 DMA requests
- Single interrupt line for multiple interrupt source events
- IIS master mode: support integer clock divider
- IIS master mode: support fraction clock divider – both variable and fixed serial clock mode
- Support 32bit width of APB 2.0 configuration.

5.13.3 Function Description

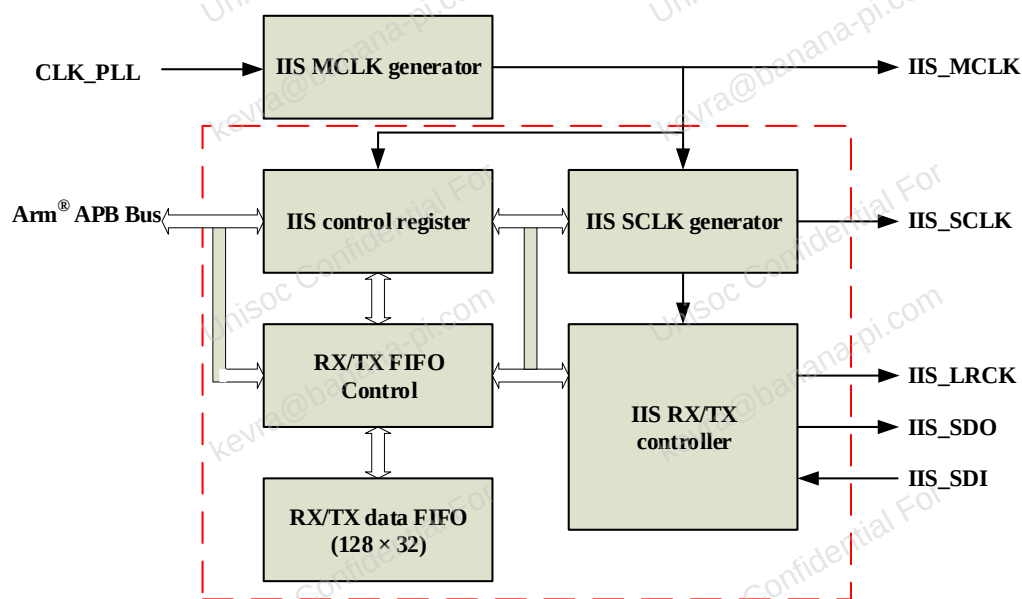
5.13.3.1 IIS interface

IIS module consists of RX/TX controller, Arm® processor APB interface control register, clock generator and FIFO control. Arm® program IIS register, enable IIS mode, starts RX/TX data transfer. Output data is written to TX FIFO and received data is stored in RX FIFO. Each FIFO is 32-bit width plus 128-address length.

MCLK generator generates main clock for IIS block. This clock also output as master clock during hand shaking. Clock divider value and control signal for MCLK come from global control register.

SCLK generator divides main clock according to a 16-bit clock divider register, which defines half SCLK clock width. It can be programmed from 0 to 32767.

Figure 5-20 IIS module diagram



IIS interface has 4 pins, IIS_SCLK is serial clock output, IIS_SDI is serial data input, IIS_SDO is serial data output, IIS_LRCK is left/right channel select. The IIS master device generates IIS_LRCK and IIS_SCLK.

Serial data is transmitted in two's complement format with the MSB first. Serial data can be sent at rising or falling edge of SCLK, at receiver, it must be latched at rising edge. When system word length is greater than the transmit word length, the word is truncated (least significant data bits are set to '0') for data transmission. If the receiver is sent more bits than its word length, the bits after the LSB are ignored. On the other hand, if the receiver is sent fewer bits than its word length, the missing bits are set to zero internally.

Three bus formats are available in this module. Following Figures illustrate the relationship between the SCLK, LRCK and serial data I/O for different interface protocols. The polarity of LRCK is programmable. The bits per channel is programmable up to 32bit.

IIS mode is where the MSB is available on the 2nd rising edge of SCLK following a LRCK transition.

Figure 5-21 IIS-compatible format

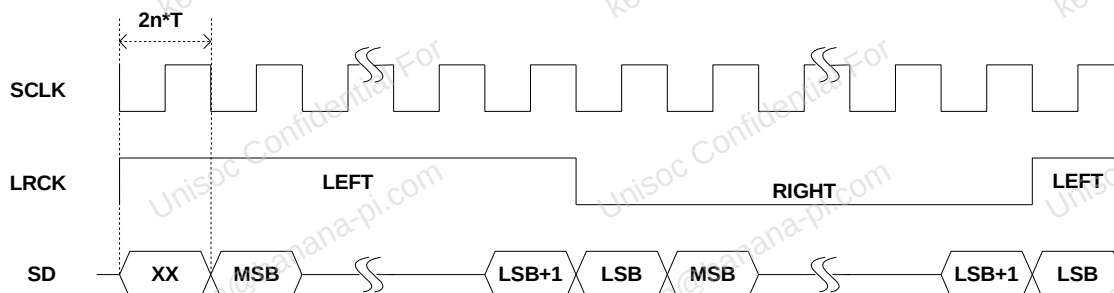
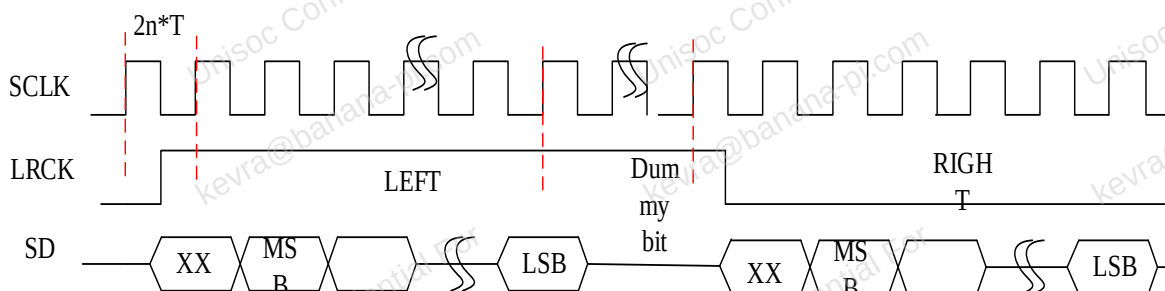


Figure 5-22 IIS-compatible format with dummy bit



MSB-justified mode is where the MSB is available on the first rising edge of SCLK following a LRCK transition.

Figure 5-23 MSB-justified format

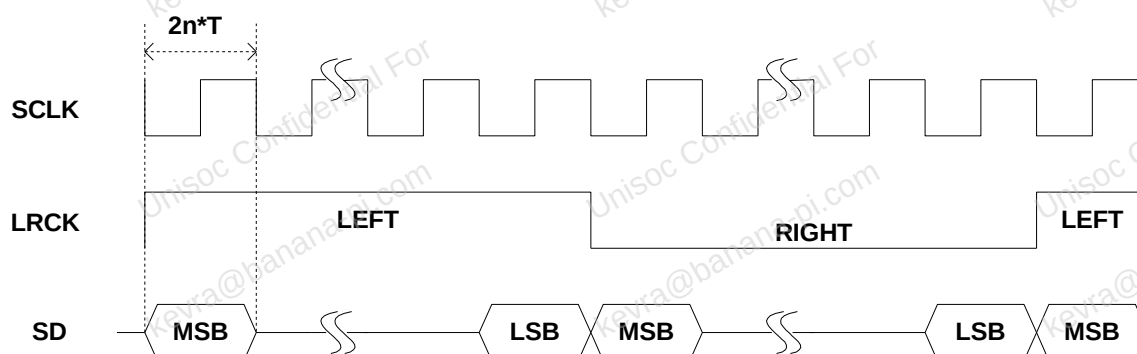
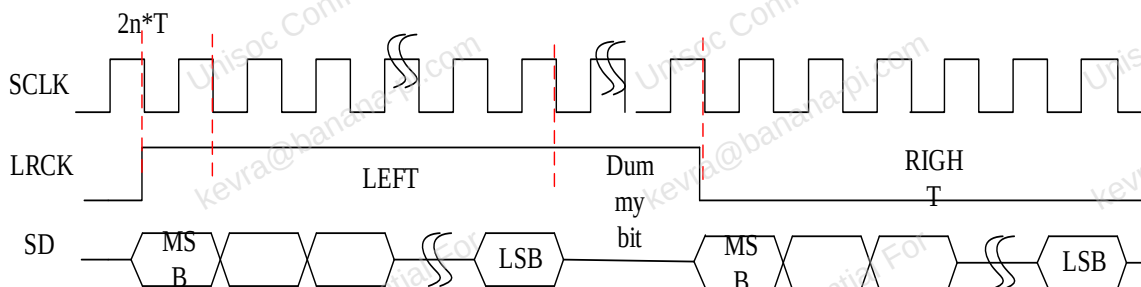
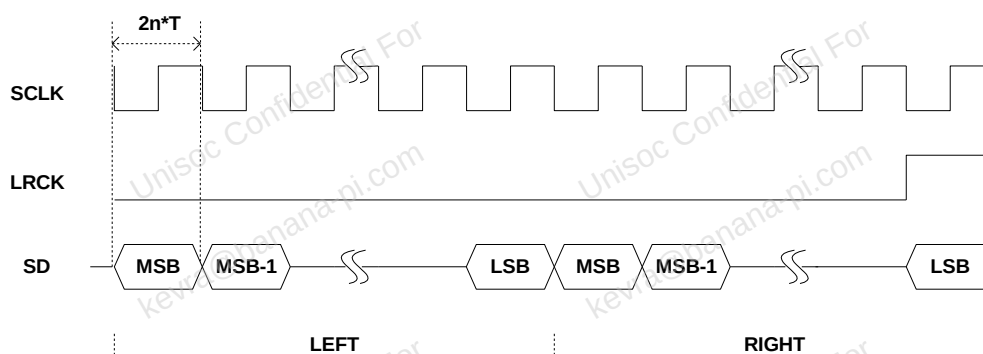


Figure 5-24 MSB-justified format with dummy bit



DSP mode is where the left channel MSB is available on either the 1st or 2nd rising edge of SCLK following a LRCK transition high. Right channel data immediately follows left channel data.

Figure 5-25 Synchronized format



NOTE

- T: 1/(system clock frequency)
- n: register programmable integer, n-1 = 0, ..., 32767

Table 5-5 IIS data format memory map

IIS_CTL bit1~0	Bit / channel	Data arrangement
00	8bit	Data on bit 31~24
01	16bit	Data on bit 31~15
1x	32bit	Data on bit 31~0

5.13.3.2 IIS work mode

5.13.3.2.1 IIS tx_only mode

IIS master can work in tx_only mode when it only transmits SCK, LRCK and SDO to slave, but doesn't capture the data from SDI. When configuring RTX_MD(IIS_CTRL0[7:6]) as 2'b10, the IIS master will work in only

transmit mode that it can't capture SDI at any moment. When configuring RTX_MD(IIS_CTRL0[7:6]) as 2'b11, the IIS master will enter tx_only mode only when tx_only_en is valid.

In this mode, the software can control the tx_soft_en (IIS_CTRL1[10]) and rx_soft_en (IIS_CTRL1[12]) to indicate the IIS master should stop to capture the SDI from slave. Software should control the next signals:

- rx_en_sel = 0 (IIS_CTRL1[13])
- rx_soft_en = 0 (IIS_CTRL1[12])
- tx_en_sel = 0 (IIS_CTRL1[11])
- tx_soft_en = 1 (IIS_CTRL1[10])

When hardware wants to control the IIS master, it should control input port iis_sink_usb_dma_enable and iis_source_usb_dma_enable. Hardware should control the next signals:

- rx_en_sel = 1 (IIS_CTRL1[13])
- iis_sink_usb_dma_enable = 0
- tx_en_sel = 1 (IIS_CTRL1[11])
- iis_source_usb_dma_enable = 1

The following Figures illustrate the relationship between the SCK, LRCK and tx_only_en for different configuration where tx_only_en = iis_tx_en & (~iis_rx_en).

Figure 5-26 IIS works in tx_only mode case1

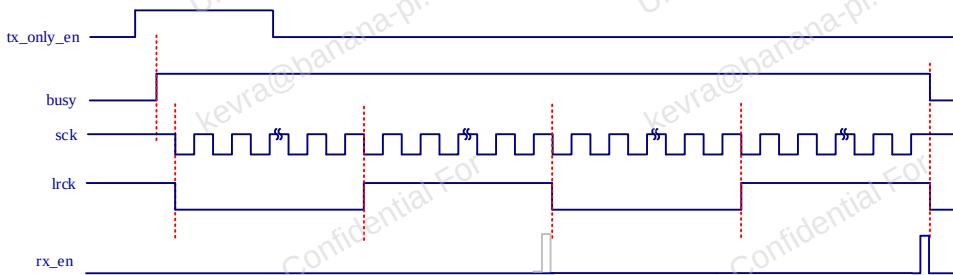
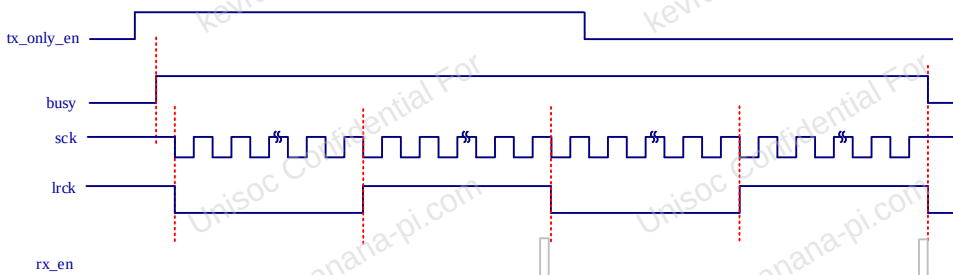


Figure 5-27 IIS works in tx_only mode case2



5.13.3.2.2 IIS rx_only mode

IIS master can work in rx_only mode when it should only generate SCK & LRCK to IIS slave and receive the serial data to rx_fifo. When configuring RTX_MD(IIS_CTRL0[7:6]) as 2'b01, the IIS master will work in only receive mode that it can't transmit at any moment. When configuring RTX_MD(IIS_CTRL0[7:6]) as 2'b11, the IIS master will enter rx_only mode only when rx_only_en is valid.

In this mode, the software can control the tx_soft_en (IIS_CTRL1[10]) and rx_soft_en (IIS_CTRL1[12]) to indicate the IIS master should start to generate the SCK & LRCK to slave. Software should control the next signals:

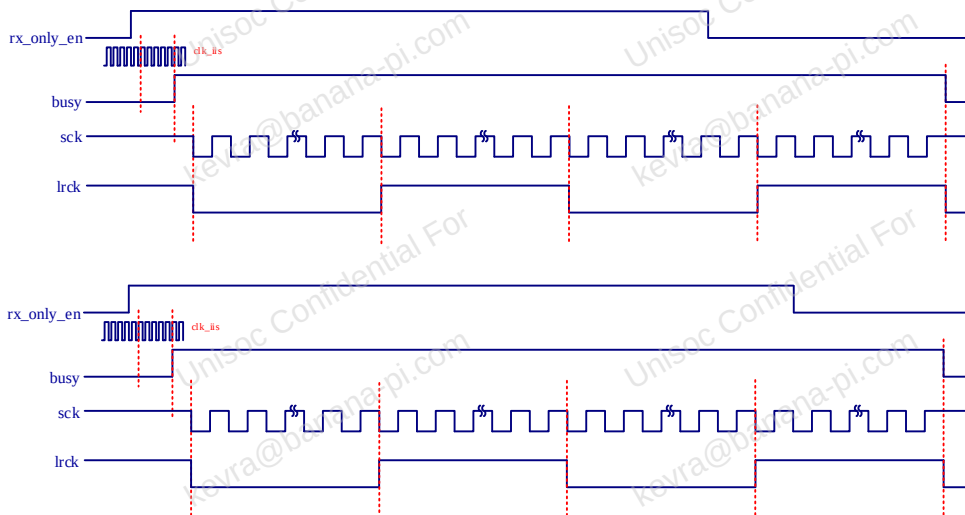
- rx_en_sel = 0 (IIS_CTRL1[13])
- rx_soft_en = 1 (IIS_CTRL1[12])
- tx_en_sel = 0 (IIS_CTRL1[11])
- tx_soft_en = 0 (IIS_CTRL1[10])

When hardware wants to control the IIS master, it should control input port iis_sink_usb_dma_enable and iis_source_usb_dma_enable. Hardware should control the next signals:

- rx_en_sel = 1 (IIS_CTRL1[13])
- iis_sink_usb_dma_enable = 1
- tx_en_sel = 1 (IIS_CTRL1[11])
- iis_source_usb_dma_enable = 0

The following Figures illustrate the relationship between the SCK, LRCK and rx_only_en for different configuration where rx_only_en = (~iis_tx_en) & iis_rx_en.

Figure 5-28 IIS works in rx_only mode



When configuring rx_only mode, IIS master will write data to RX_FIFO after capturing the left/right channel's data. For there will be 2 cycles for synchronizing, the rx_only_en should not be pulled down at the nearby of the right channel end.

The cycles between rx_only_en and busy will about 7, where 2 cycles are used for synchronizing APB domain to clk_iis domain and the other 5 cycles are fixed delay.

5.13.3.2.3 IIS rtx mode

IIS master work in rtx mode default, where it generates SCK & LRCK to IIS slave and receives the serial data from SDI to rx_fifo. It should configure RTX_MD(IIS_CTRL0[7:6]) as 2'b11.

In this mode, the software should control the next signals:

- rx_en_sel = 0 (IIS_CTRL1[13])

- rx_soft_en = 1 (IIS_CTRL1[12])
- tx_en_sel = 0 (IIS_CTRL1[11])
- tx_soft_en = 1 (IIS_CTRL1[10])

When hardware wants to control the IIS master, it should control input port iis_sink_usb_dma_enable and iis_source_usb_dma_enable. Hardware should control the next signals:

- rx_en_sel = 1 (IIS_CTRL1[13])
- iis_sink_usb_dma_enable = 1
- tx_en_sel = 1 (IIS_CTRL1[11])
- iis_source_usb_dma_enable = 1

NOTE

When iis_rx_en is valid, the IIS master will always generate SCK & LRCK to IIS slave regardless of the fact that tx_fifo is already empty. In this case, the situation is the same as rx_only mode.

Based on the above, the software should configure the {rx_soft_en, tx_soft_en} as 2'b00 before it fill the data to tx_fifo. When tx_fifo is filled with data, configuring the {rx_soft_en, tx_soft_en} as 2'b11 to begin the data transfer.

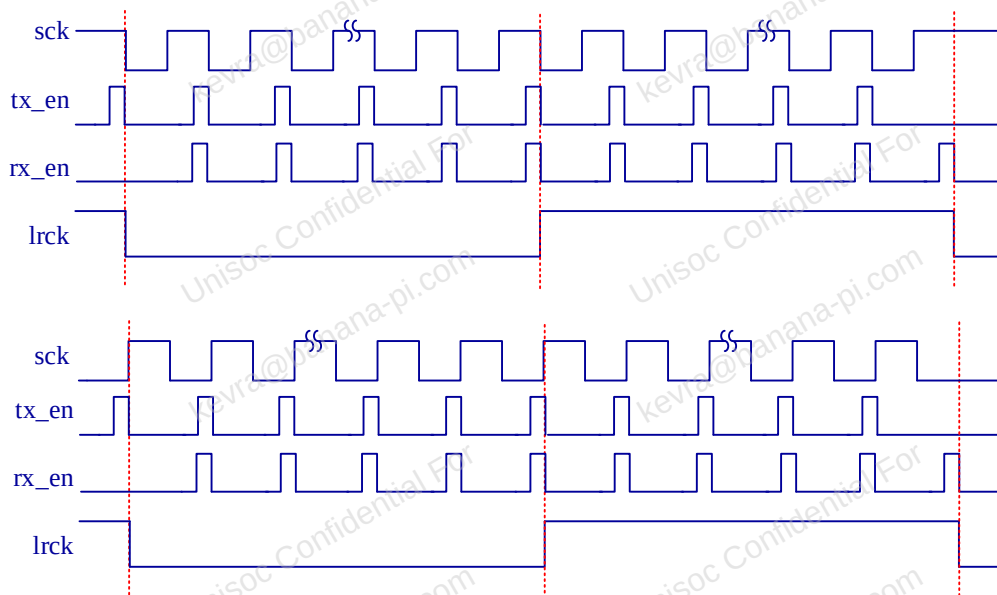
5.13.3.3 IIS TX/RX Edge

- The clock edge of TX data shift out can be configured by NG_TX (IIS_CTRL0[1]).
 - NG_TX = 1, nededge of SCK
 - NG_TX = 0, posedge of SCK
- The clock edge of RX data shift in can be configured by NG_RX (IIS_CTRL0[0]).
 - NG_RX = 1, nededge of SCK
 - NG_RX = 0, posedge of SCK

SCK_REV (CTRL0[11]) can be configured to indicate whether SCK inverse enable or not.

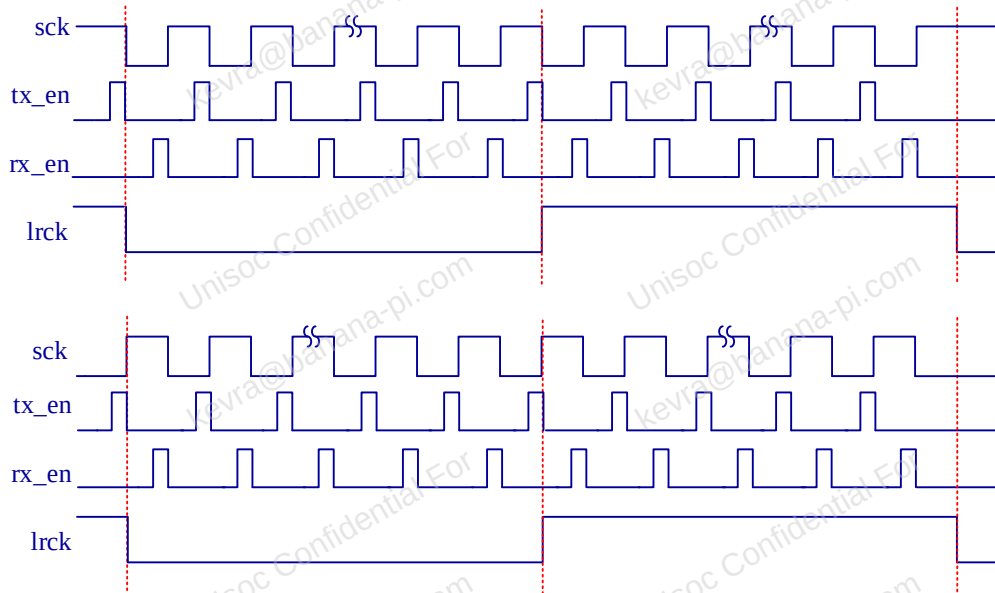
When configuring {NG_TX, NG_RX} = 2'b00 or 2'b11, it means there will one full sck timing to receive the data from slave.

Figure 5-29 {NG_RX, NG_TX} is 2'b00 or 2'b11



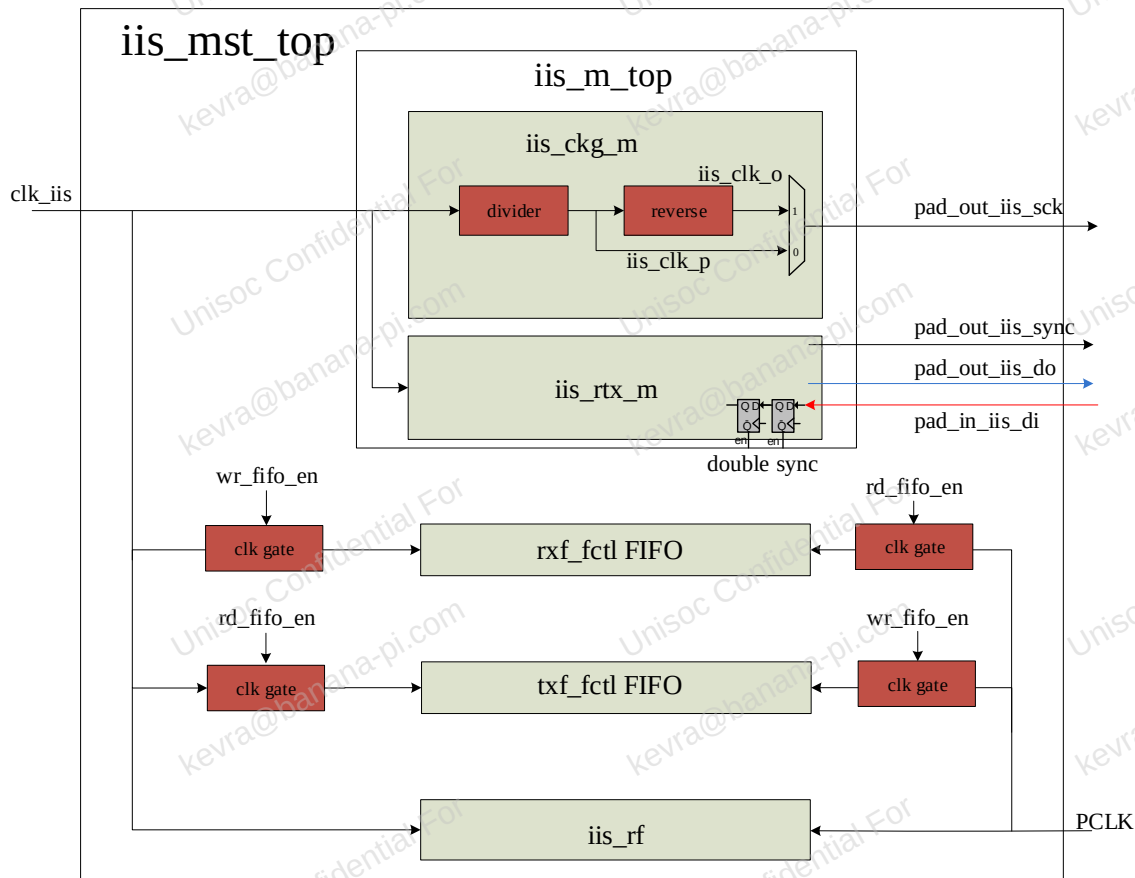
When configuring $\{NG_TX, NG_RX\} = 2'b01$ or $2'b10$, it means there will only half sck timing to receive the data from slave.

Figure 5-30 $\{NG_RX, NG_RX\}$ is $2'b01$ or $2'b10$



5.13.3.4 IIS clock path

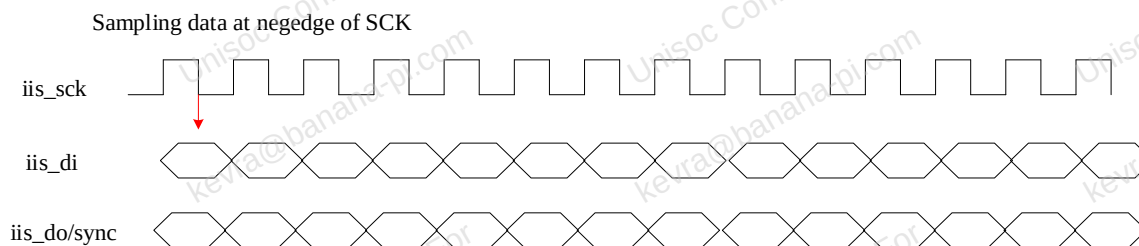
Figure 5-31 IIS clock path



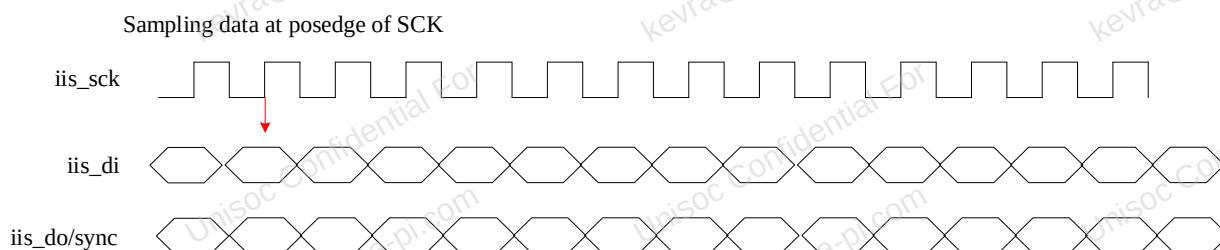
- Module clock: iis_clk, PCLK.
- IO pad: iis_sck, iis_sync, iis_di, iis_do.
- IIS_MST can only be configured as master mode.
- “divider” block function: $iis_clk_p = iis_clk / M$. where M is a clock divider factor.
- “reverse” block function: $iis_clk_o = \sim iis_clk_p$.
- “sck counter” block function: count iis_sck period. Count unit is iis_clk.
- “double sync” block is used to resolve crossing clock domain signal.

5.13.3.5 Pad IO timing

In default mode, IIS output signals like iis_do, iis_sync signal are sent at the posedge of iis_sck. And input signals like iis_di are sampled at the negedge of iis_sck.

Figure 5-32 Sampling data at negedge of IIS_SCK


You can also set output signal sent edge at negedge and sampling edge at posedge, like the following.

Figure 5-33 Sampling data at posedge of IIS_SCK


The implementation of sampling edge is as following.

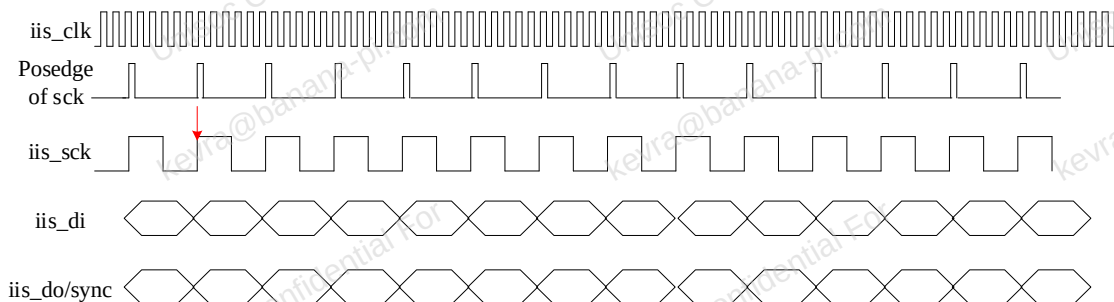
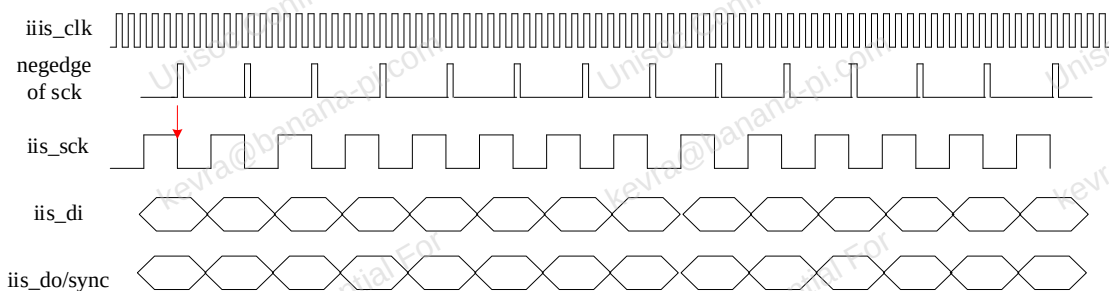
Figure 5-34 implementation of sampling data at negedge of IIS_SCK


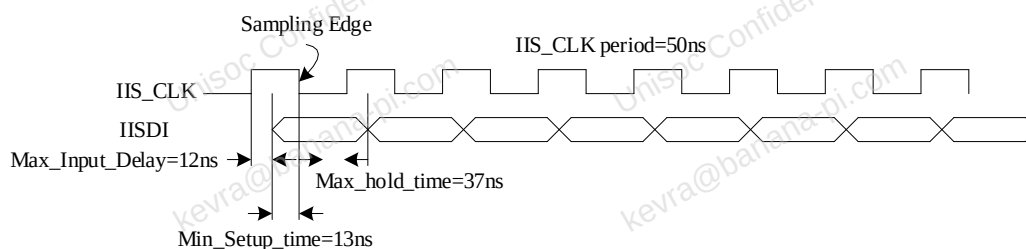
Figure 5-35 implementation of sampling data at posedge of IIS_SCK



5.13.3.6 Setup time and hold time

If assume max input delay is 10ns and min input delay is 0ns for iis_di to iis_sck. max output delay is 10ns and min output delay is 5ns for iis_do/iis_lrck to iis_sck. The clock period time is 50ns for iis_sck.

Figure 5-36 implementation of sampling data at posedge of IIS_SCK



Then for iis_di, the setup time range is $13\text{ns} < \text{setup_time} < 25\text{ns}$, and the hold time range is $25\text{ns} < \text{hold_time} < 37\text{ns}$.
For iis_do/iis_lrck, the setup time range is $13\text{ns} < \text{setup_time} < 25\text{ns}$, and the hold time range is $25\text{ns} < \text{hold_time} < 37\text{ns}$.

5.13.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.13.5 Application Notes

5.13.5.1 IIS Program examples

- Justified format, master, 16 bits per-channel, LRCK low for left, Program control register:
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_08d1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0000
- IIS format, master, 8 bits per-channel, LRCK high for left, Program control register:
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_0dc1
 - iis_ctl1 = 32'h0000_0f11

- iis_ctl2 = 32'h0000_0000
- 3. DSP (sync) format, master, 16 bits per-channel, MSB is available on the 1st rising edge, Program control register
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_0ad1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0000
- 4. DSP format, master, 16 bits per-channel, MSB is available on the 1st rising edge, Program control register:
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_0ed1
 - iis_ctl1 = 32'h0000_3f11
 - iis_ctl2 = 32'h0000_7800

5.13.5.2 Clock dividing modes

In master mode, the IIS serial clock iis_sck is sent from the IIS module, and this clock is divided from clk_iis. IIS module provides three modes for clock dividing.

- Integer clock dividing mode
 - This mode can be used only if iis_sck is integrally divided from clk_iis.
 - Mode enable: IIS_CLK_DVDR_MD0 = 0 and IIS_CLK_DVDR_MD1 = 0
 - Only IIS_CLKD is used for dividing; IIS_CLKM and IIS_CLKN are DON'T CARE.
 - $IIS_CLKD = (clk_iis / (2 \times iis_sck)) - 1$; $iis_sck = \text{sample rate} \times \text{channel number} \times \text{channel length}$

For example, if clk_iis runs at 153.6MHz, and IIS module is configured as 48kHz sample rate, 2 channel and 16-bit channel length:

 - iis_sck should be $48k \times 2 \times 16 = 1.536\text{MHz}$
 - IIS_CLKD should be $(153.6\text{M} / (2 \times 1.536\text{M})) - 1 = 49$
- Fraction clock dividing mode0
 - In this mode, the actual frequency of iis_sck is variable, and only average is target frequency. Because of this variability, the iis_lrck frequency is also variable, and only average is target sample rate.
 - Mode enable: IIS_CLK_DVDR_MD0 = 1 and IIS_CLK_DVDR_MD1 = 0
 - IIS_CLKM and CLKN are used for dividing; while IIS_CLKD is not cared.
 - $IIS_CLKM / IIS_CLKN = 2 \times (\text{target } iis_sck) / clk_iis$; $(\text{target } iis_sck) = \text{sample rate} \times \text{channel number} \times \text{channel length}$
 - $IIS_CLKM = 2 \times \text{sample rate (Hz)} \times \text{channel number} \times \text{channel length} / 100$
 - $IIS_CLKN = clk_iis \text{ (Hz)} / 100$

For example, if clk_iis runs at 153.6MHz, and IIS module is configured as 44.1kHz sample rate, 2 channel and 16-bit channel length:

 - $IIS_CLKM = 2 \times 44100 \times 2 \times 16 / 100 = 28224$
 - $IIS_CLKN = 153600000 / 100 = 1536000$
- Fraction clock dividing mode1
 - In this mode, the frequency of iis_sck is fixed. But the actual sample rate is variable, and only average is target sample rate.
 - Compared with MD0, the advantage of MD1 is that iis_sck is fixed, but the disadvantage is that the jitter of sample rate is higher.

- Mode enable: IIS_CLK_DVDR_MD0 = 0 and IIS_CLK_DVDR_MD1 = 1
- IIS_CLKM, CLKN and CLKD are used for dividing.
- $IIS_CLKD = (clk_iis / (2 \times iis_sck)) - 1$
- $IIS_CLKM / IIS_CLKN = (target\ sample\ rate) \times channel\ number \times channel\ length / iis_sck$; IIS_CLKM must be not more than IIS_CLKN.
- $IIS_CLKM = 2 \times (IIS_CLKD + 1) \times (target\ sample\ rate) \times channel\ number \times channel\ length / 100$
- $IIS_CLKN = clk_iis / 100$
- The iis_sck (or IIS_CLKD) should be decided by software. iis_sck must be not lower than ((target sample rate) $\times$ channel number $\times$ channel length). And iis_sck should be not too high for timing issue. Usually iis_sck is a little bit higher than ((target sample rate) $\times$ channel number $\times$ channel length).

For example, if clk_iis runs at 153.6MHz, and IIS module is configured as 44.1kHz sample rate, 2 channel and 16-bit channel length. Firstly, choose iis_sck as $153.6M / (2 \times (44 + 1)) = 1.706667MHz$ because ((target sample rate) $\times$ channel number $\times$ channel length) = 1.4112MHz and 1.706667MHz is a little bit higher than 1.4112MHz.

- $IIS_CLKD = 44$
- $IIS_CLKM = 2 \times (44 + 1) \times 44100 \times 2 \times 16 / 100 = 1270080$
- $IIS_CLKN = 153600000 / 100 = 1536000$

Some guidelines for clock dividing modes:

- Integer clock dividing mode is preferred if possible, because this mode has fixed serial clock frequency and fixed sample rate.
- If iis_sck cannot be integrally divided from clk_iis, choose fraction clock dividing modes. If system needs fixed serial clock frequency, mode1 is chosen. Otherwise mode0 is preferred for small sample rate jitter.
- The frequency options of clk_iis depend on different chips. But 153.6MHz is always preferred because this frequency can be divided integrally to support 48k, 24k, 12k, 32k, 16k and 8k sample rate.
- Fraction clock divide mode1 configure.

Table 5-6 IIS configure for fraction clock divide

MD0, MSB-justified	Master	Slave
IIS_CTL[5]	0	x(don't care)
IIS_CTL[4]	1	x(don't care)
IIS_CTL[1]	x(don't care)	x(don't care)
IIS_CTL[0]	x(don't care)	x(don't care)

MD0, compatible	Master	Slave
IIS_CTL[5]	0	x(don't care)
IIS_CTL[4]	1	x(don't care)
IIS_CTL[1]	x(don't care)	x(don't care)
IIS_CTL[0]	x(don't care)	x(don't care)

MD1, MSB-justified	Master	Slave
IIS_CTL[5]	1	x(don't care)
IIS_CTL[4]	0	x(don't care)
IIS_CTL[1]	x(don't care)	0
IIS_CTL[0]	x(don't care)	1

MD1, compatible	Master	Slave
IIS_CTL[5]	1	x(don't care)
IIS_CTL[4]	0	x(don't care)
IIS_CTL[1]	x(don't care)	1
IIS_CTL[0]	x(don't care)	0

Table 5-7 iis_clk configure suggestion

IIS mode	rate	width	clk_iis(Hz)
compatible	192k	32	153.6M(Suggest)
compatible	192k	32	128M(Suggest)
compatible	192k	32	80M(Not Suggest)
compatible	192k	32	26M(Not Suggest)
compatible	192k	16	153.6M(Suggest)
compatible	192k	16	128M(Suggest)
compatible	192k	16	80M(Suggest)
compatible	192k	16	26M(Not Suggest)
compatible	192k	8	153.6M
compatible	192k	8	128M
compatible	192k	8	80M
compatible	192k	8	26M
compatible	44.1k	32	153.6M
compatible	44.1k	32	128M
compatible	44.1k	32	80M
compatible	44.1k	32	26M

IIS mode	rate	width	clk_iis(Hz)
compatible	44.1k	16	153.6M
compatible	44.1k	16	128M
compatible	44.1k	16	80M
compatible	44.1k	16	26M
compatible	44.1k	8	153.6M
compatible	44.1k	8	128M
compatible	44.1k	8	80M
compatible	44.1k	8	26M
Justified	192k	32	153.6M(Suggest)
Justified	192k	32	128M(Suggest)
Justified	192k	32	80M(Not Suggest)
Justified	192k	32	26M(Not Suggest)
Justified	192k	16	153.6M(Suggest)
Justified	192k	16	128M(Suggest)
Justified	192k	16	80M(Suggest)
Justified	192k	16	26M(Not Suggest)
Justified	192k	8	153.6M
Justified	192k	8	128M
Justified	192k	8	80M
Justified	192k	8	26M
Justified	44.1k	32	153.6M
Justified	44.1k	32	128M
Justified	44.1k	32	80M
Justified	44.1k	32	26M
Justified	44.1k	16	153.6M
Justified	44.1k	16	128M
Justified	44.1k	16	80M
Justified	44.1k	16	26M
Justified	44.1k	8	153.6M
Justified	44.1k	8	128M
Justified	44.1k	8	80M

IIS mode	rate	width	clk_iis(Hz)
Justified	44.1k	8	26M

NOTE

In IIS fraction clock divide model, clk_iis frequency should be higher when bit clock is fast.

5.13.5.3 Change list with IIS

IIS_MST will be different with IIS when it is configured as 32 bits mode. The difference will happen when tx_fifo is filled with an odd number of data. The IIS will only transmit the last left channel data, and won't transmit the right channel data. But IIS_MST will transmit an extra right channel data; the data is same as the left channel data.

So suggest software should ensure tx_fifo filled with an even number of data when it tests the function of 32bits mode. The waveform is as follows.

Figure 5-37 iis_mst (tx_fifo filled with 3 data)

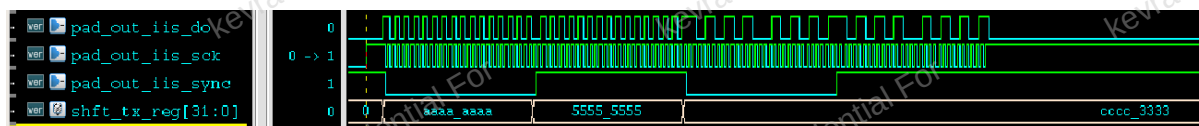
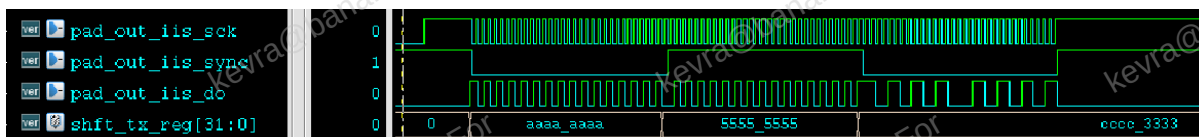


Figure 5-38 iis (tx_fifo filled with 3 data)



5.13.5.4 Debug tips

- To ensure that it is working well, the first step is to test iis_sck, iis_lrck, iis_di, iis_do using scope or by observing state of IIS_STS2.
- In the IIS mode, iis_lrck changes periodically, and high/low duty ratio is 50%.
- If iis_sck, iis_lrck, iis_di, iis_do is normal, then you can operate IIS_TXD. Transmitting a word by writing data to IIS_TXD in one side(e.g. transmit). Observe the result by reading data from IIS_TXD in another side(e.g. receiver).

5.14 INTC

5.14.1 Overview

Interrupt Controller generate IRQ (Interrupt Request) and FIQ (Fast Interrupt Request) to Arm® processor from 32 IRQ and 32 FIQ channels.

5.14.2 Features

- Provide 30 hardware-triggered IRQ channels from IRQ channel 2 to channel 31
- Provide 30 hardware-triggered FIQ channels from FIQ channel 2 to channel 31

- Provide 1 software-triggered IRQ channel from IRQ channel 1
- Provide 1 software-triggered FIQ channel from FIQ channel 1
- Provide 1 special IRQ channel from IRQ channel 0
- Provide 1 special FIQ channel from FIQ channel 0
- Each IRQ or FIQ channel can be enabled or disabled independently
- Provide raw status for each IRQ or FIQ channel
- Provide masked status for each IRQ or FIQ channel
- Provide software debug mode to perform interrupt test

5.14.3 Function Description

- Interrupt Controller generate IRQ to Arm® processor from 32 IRQ channels, including 30 hardware-triggered IRQ channels, 1 special hardware-triggered IRQ channel and 1 software-triggered IRQ channel.
- Interrupt Controller also generate FIQ to Arm® processor from 32 FIQ channels, including 30 hardware-triggered FIQ channels, 1 special hardware-triggered FIQ channel and 1 software-triggered FIQ channel.
- 32 IRQ channels are independent from 32 FIQ channels; they can be connected or configured independently. Following is the function diagram of Interrupt Controller.

Figure 5-39 Function diagram

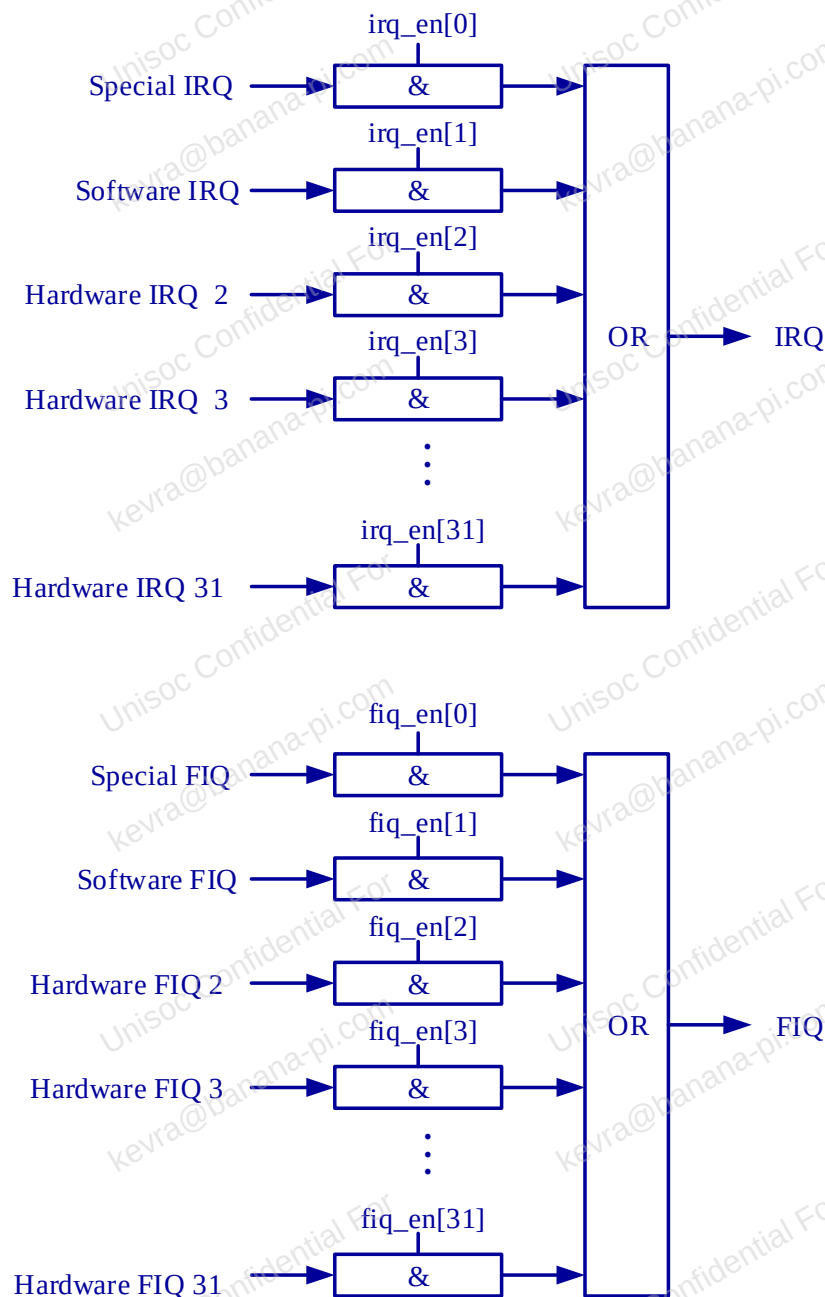
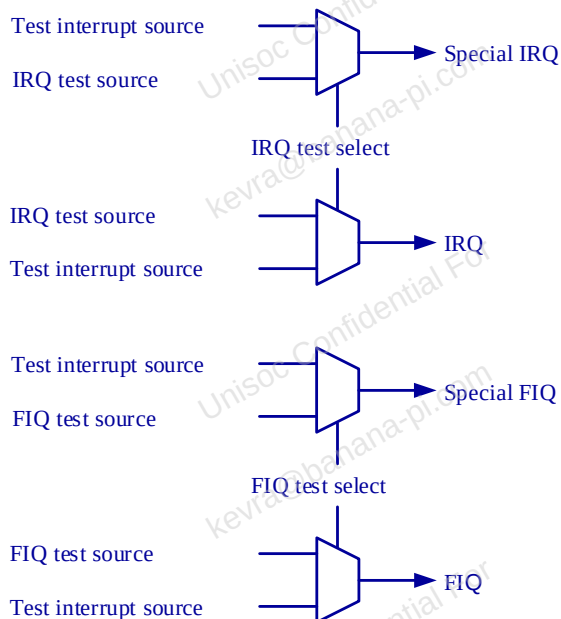


Figure 5-40 Interrupt source



5.14.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.14.5 Application Notes

Make sure that all the hardware interrupts are level triggered, and input level is active high. Once interrupt is generated, it should keep high until software clear the interrupt. The following figure is a recommended flow for hardware channels.

Software interrupt channel could be used to specify a flag to INTC on APB interface, then detect it as a interrupt.

5.15 KPD Module

5.15.1 Overview

Here Keypad means “keypad controller” which is one APB device in Arm[®] processor . The keypad controller scans the external “key matrix” whose maximum size is 8 rows x 8 columns. It scans the “key matrix” row by row and checks the keys of the row one by one. There are at most four keys “pressed”. “released” status can be detected in the same time, which is the so called “multi-key detection”. On the other hand, one pressed key can generate multi interrupts periodically to software and this mode is named the “long key” mode.

For the reason that the external “key matrix” is pure mechanical, if the geometric shape constituted by any three “pressed keys” is a “right-angled triangle”, the keypad controller will detect another “inductive pressed” key. For example, if key 1 (x1, y1), key 2 (x2, y2), key 3 (x1, y2) are pressed then the key 4 (x2, y1) will also be detected as the “pressed” key. The keypad controller will check the geometric shape constituted by the three “pressed keys” and if above condition is detected one “error” status will be reported to software, which should discard current error multi-keys combination.

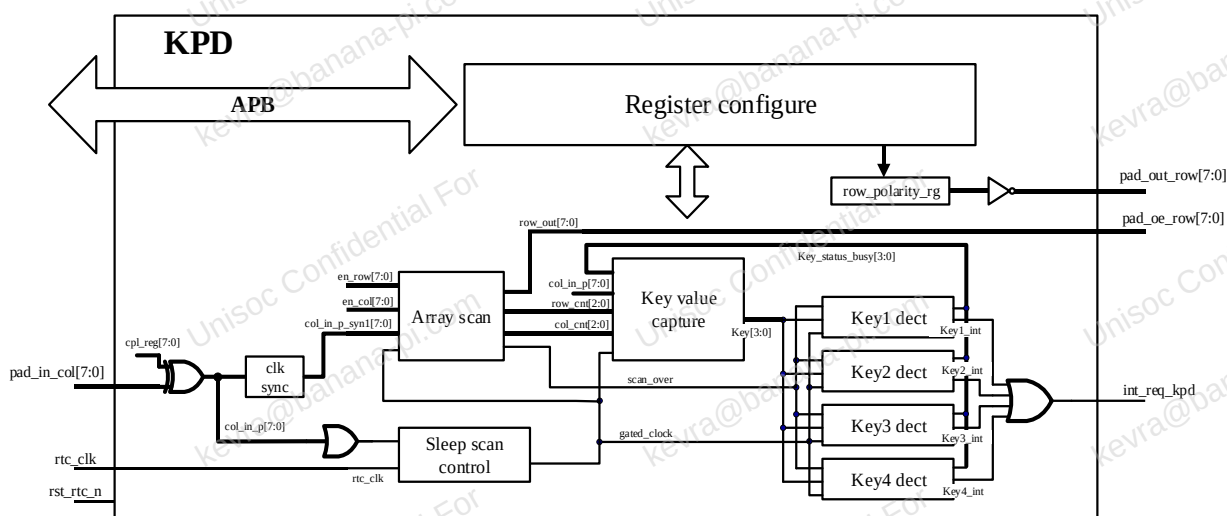
Finally, low power of the keypad controller is very important. If the controller finds that none key is pressed in the specific time, the clock of most logic inside the controller will be disabled to save power and controller goes into the sleep mode. In the sleep mode controller will be sensitive to any change of the “key matrix”. In other words, any key’s press or release will wakes up the controller to scan the “key matrix”.

5.15.2 Features

- Support maximum 8 row $\times$ 8 column key matrix and size is programmable
- Support at most four keys detection under “multi-key” mode
- Support “long key” or “single key” mode for any pressed key
- Support “right-angled triangle” geometric shape detection for the pressed keys
- Support sleep mode to save power
- Support programmable “debounce” time for key’s press and release
- Support programmable I/O polarity
- Support programmable scan speed
- Support detecting GPIO connection with one row multi columns

5.15.3 Function Description

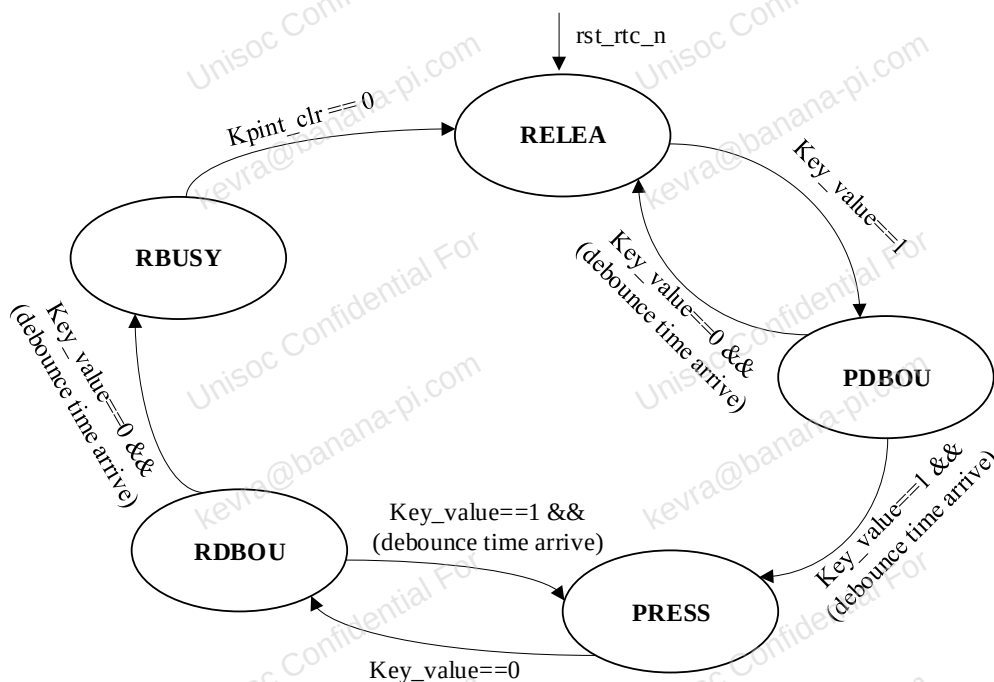
Figure 5-41 Keypad function diagram



The FSM is used to detect the key press, release and long key press, the initial state is in the RELEA. Each state will be explained in the follow:

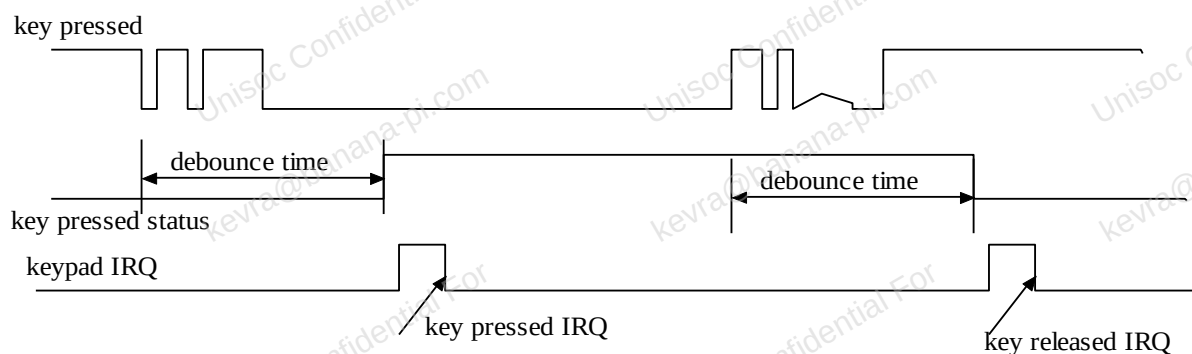
- RELEA: the key state in the idle
- PDBOU: detected key press after stable a debounce time, the state will jump to relea or press
- PRESS: key is in the state of press
- RDBOU: detected key release after stable a debounce time, the state will jump to rbusy or press
- RBUSY: the key release int generated and will jump to relea until keypad int clear

Figure 5-42 Key status logic state machine



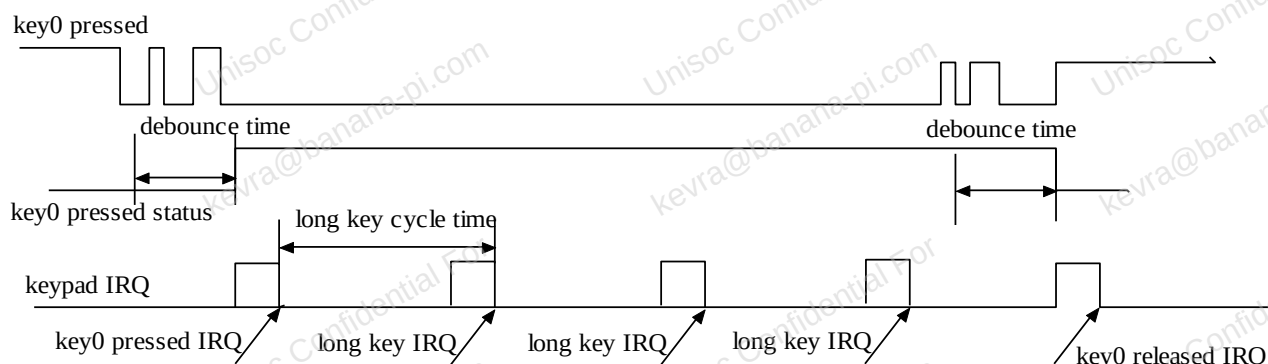
When a single button is pressed, after the debounce time, the key press int will be assert, similar with key press, key release int will be assert when key state from press to release.

Figure 5-43 Single-key mode



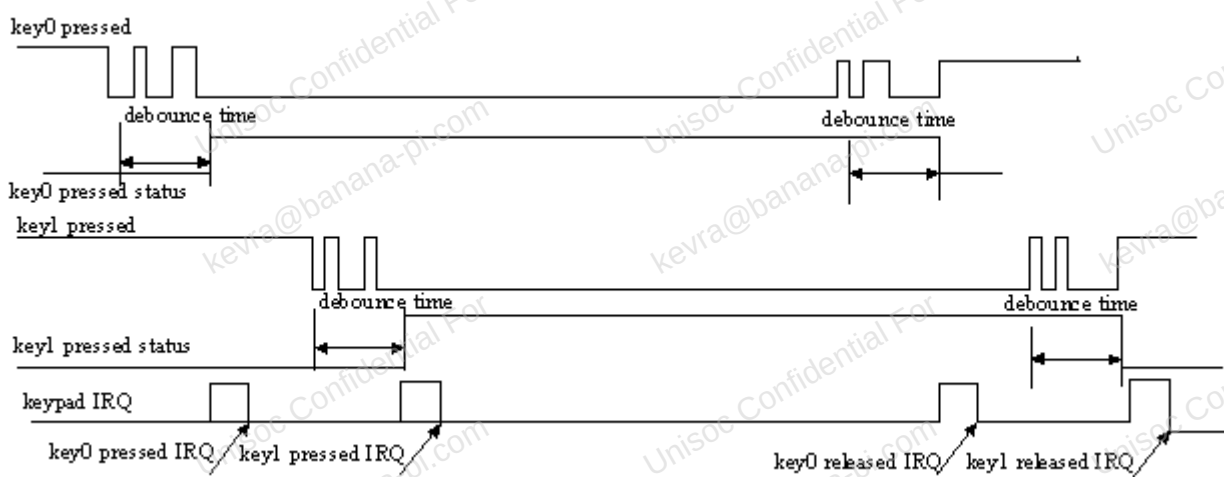
If the long key mode enable, the key pressed more than the specified time, one pressed key can generate multi long key interrupts periodically to software until the key released.

Figure 5-44 Long-key mode



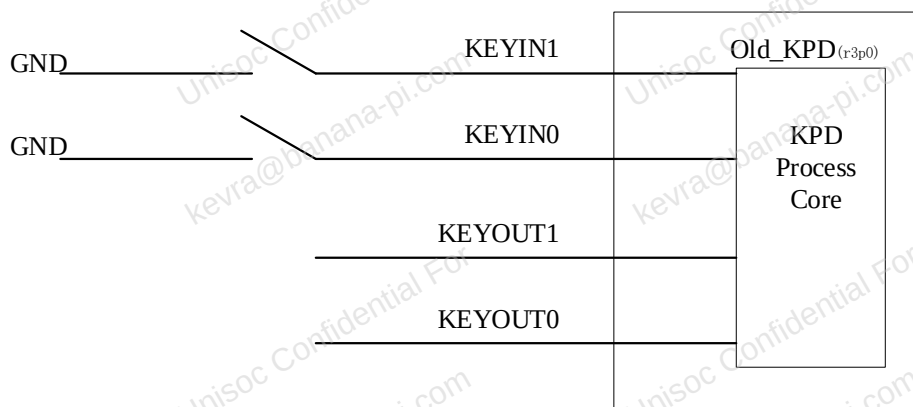
The scan time of each key is not overlapping, so the interruption of each key is independent.

Figure 5-45 Multi-key mode



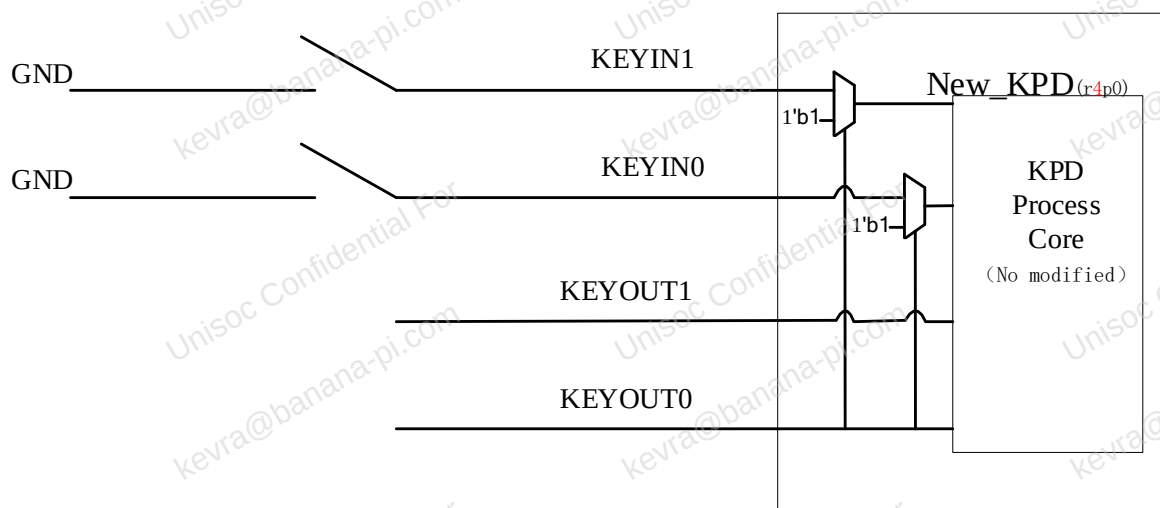
In order to reduce chip PAD, GPIO only uses one switch to connect Keyin0 and GND. Row scan output signal Keyout is not used. If r3p0 version KPD works in GPIO connection showing as following. r3p0 version KPD will detect two keys pressed, but there is only one key is pressed actually. If you press two keys, it shows four keys was pressed.

Figure 5-46 GPIO connection with old KPD (r3p0)



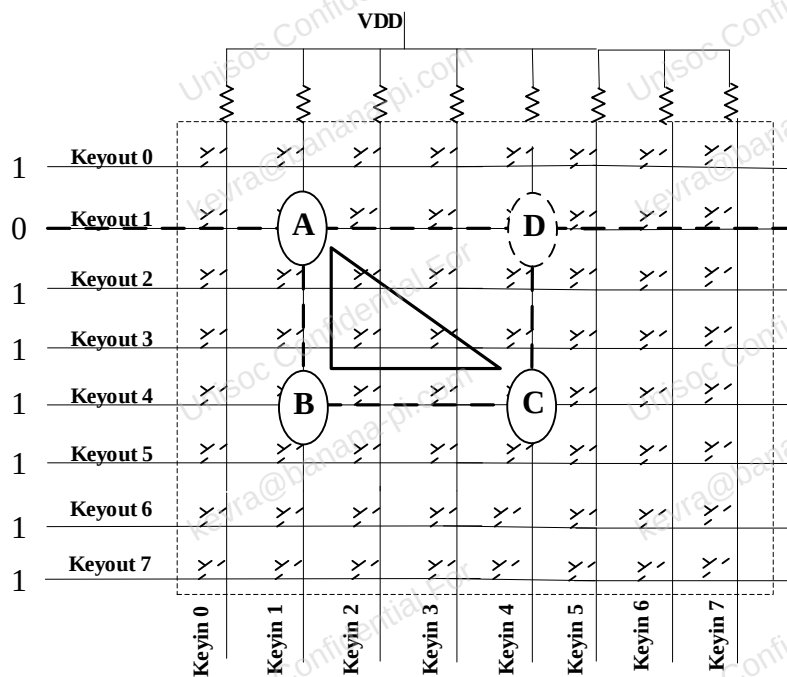
In order to support detecting one row multi columns GPIO connection, increase eight MUXs in KPD IP (each column increases one MUX), shown as [Figure 5-47](#). This makes each column Keyin signal is effective only when row scan Keyout signal is effective, so that the new version KPD can support one row multi columns GPIO connection.

Figure 5-47 r4p0 KPD modification strategy



The key A, key B, key C are pressed then the key D will also be detected as the “pressed” key. The keypad controller will check the geometric shape constituted by the three “pressed keys” and if above condition is detected one “error” status will be reported to software.

Figure 5-48 Keypad detected one “error” status

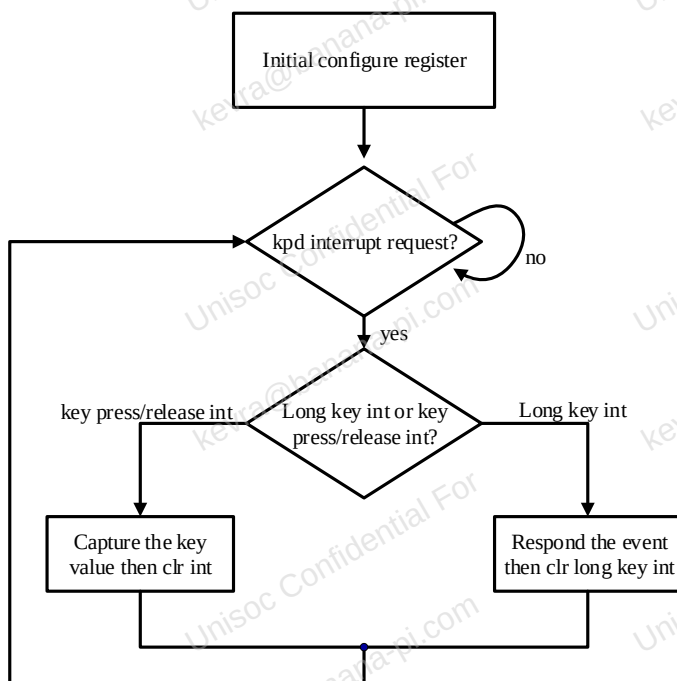


5.15.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.15.5 Application Notes

Figure 5-49 software control flow



Initial the configure registers:

1. Set the int enable (KPD_INT_EN: 0x0004).
2. Configure counter register, you can also use the default value (KPD_DEBOUNCE_CNT: 0x001c; KPD_LONGKEY_CNT: 0x0020; KPD_SLEEP_CNT: 0x0024; KPD_CLK_DIVIDE_CNT: 0x0028).
3. Configure the polarity register, depend on the physical connection (KPD_POLARITY: 0x0018).
4. Load an initial value to counter (TMR_LOAD: 0x0204).
5. Set the kpd enable (KPD_CTRL: 0x0000).

NOTE

- Support four key detect means kpd can detected four key press event at the same time.
- Kpd_four_key_cross is used to detect an error status. The right-angled triangle require the 3 key press be in the same row and the same col.
- Key status register assigned by the key detect order.

5.16 MailBox

5.16.1 Overview

MailBox is the control logic function for management of Jobs between AP/CP Processors. Mailbox's main receive mails contain some control information from different processors, and send those mails to the right target, it will send proper IRQ to target/source processors to notice them the status about mails.

5.16.2 Features

- Source processor write mail into InBox.
- Mail Controller delivers the mail into Target processor Box
- Box number depends on parameter(1~16)
- OutBox store mails in FIFO (SRAM)
- Each OutBox's FIFO depth can be set by register file independently(1~256)
- Debug mode supported, able to access all mail by debug pointer
- Two working mode: FIFO mode, IRQ mode
- Target OutBox sends out interrupt to Target processor if FIFO is not empty
- Source InBox sends out interrupt to Source processor if mail is send, block or overflow

5.16.3 Function Description

Figure 5-50 Mail Box top diagram

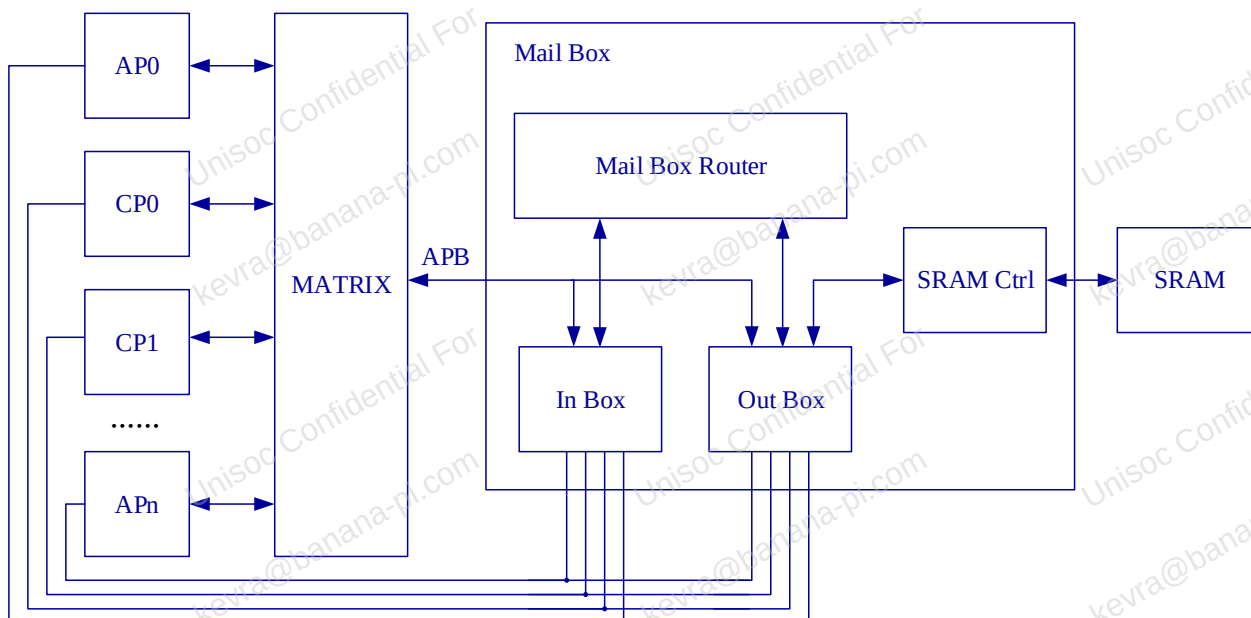
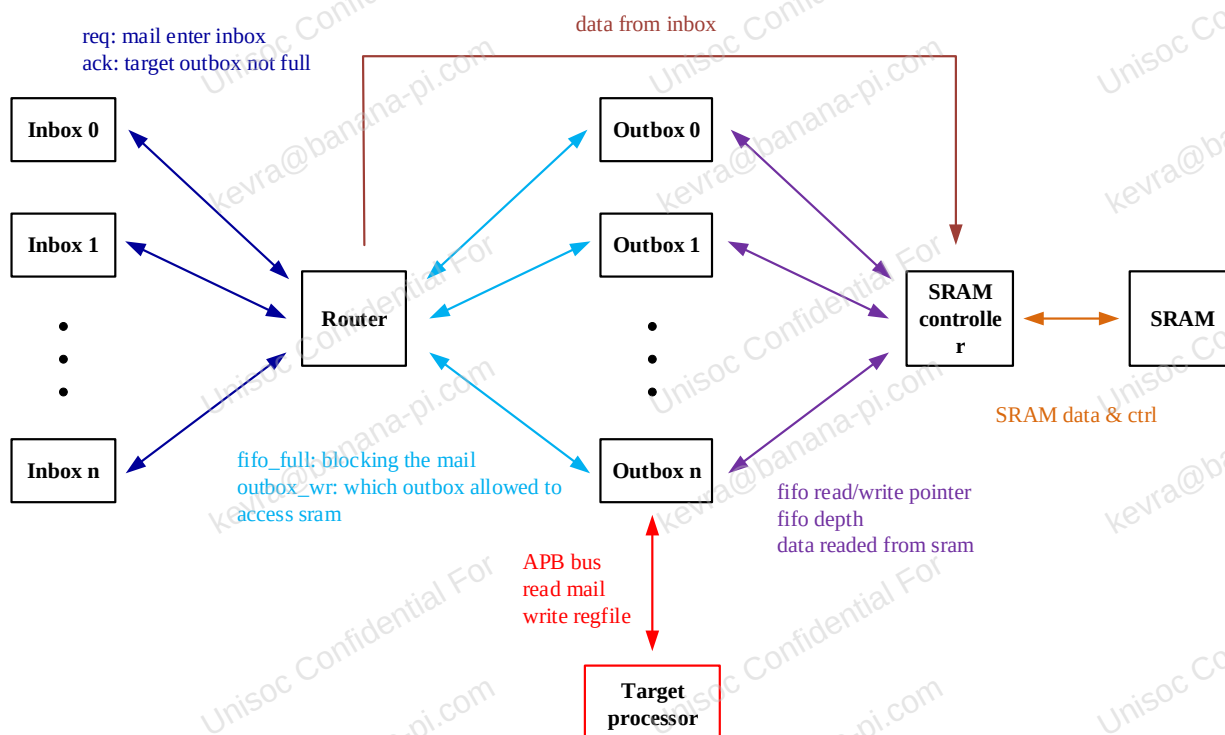
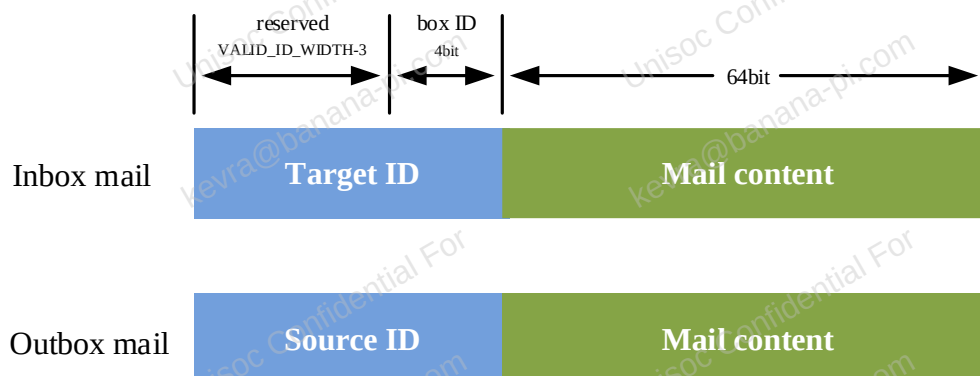


Figure 5-51 Mail Box data flow



5.16.3.1 Mail Description

Figure 5-52 Mail description



Target nbit

Source nbit, n: depends on parameter VALID_ID_WIDTH, default 8

Content 64bit

- Box_ID[3:0]=4'b0000: box[0];
- Box_ID[3:0]=4'b0001: box[1];
- Box_ID[3:0]=4'b0010: box[2];

- Box_ID[3:0]=4'b0011: box[3];
- Box_ID[3:0]=4'b0100: box[4];
- Box_ID[3:0]=4'b0101: box[5];
- Box_ID[3:0]=4'b0110: box[6];
- Box_ID[3:0]=4'b0111: box[7];
- Box_ID[3:0]=4'b1000: box[8];
- Box_ID[3:0]=4'b1001: box[9];
- Box_ID[3:0]=4'b1010: box[a];
- Box_ID[3:0]=4'b1011: box[b];
- Box_ID[3:0]=4'b1100: box[c];
- Box_ID[3:0]=4'b1101: box[d];
- Box_ID[3:0]=4'b1110: box[e];
- Box_ID[3:0]=4'b1111: box[f];

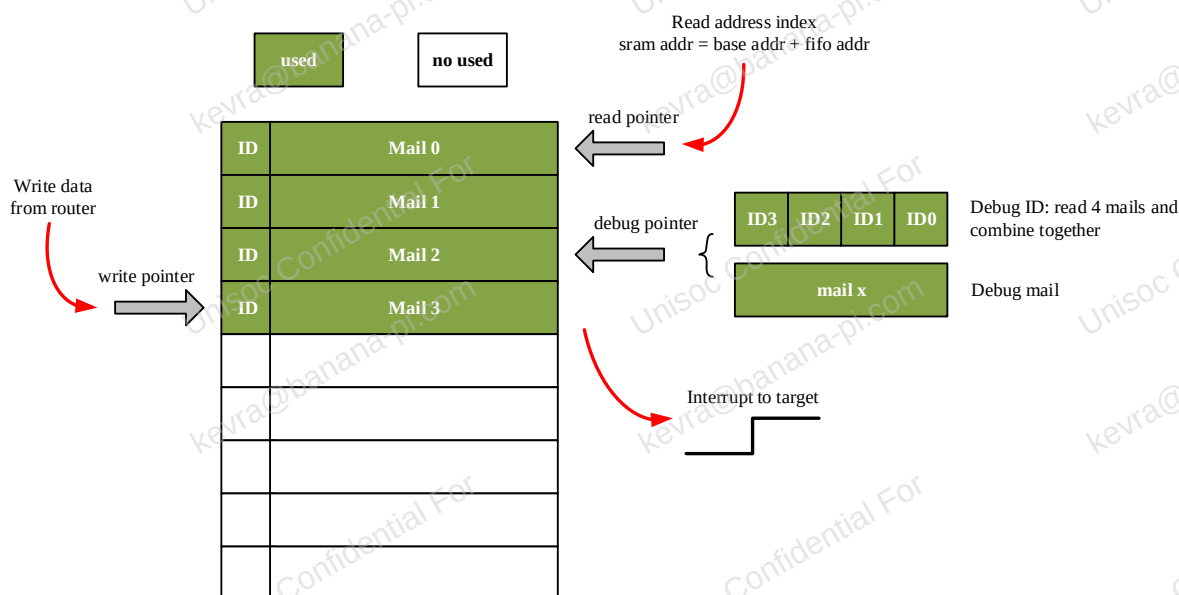
Since the max box number are 16, only use the lower 4 bit of target/source ID to decode box select signal, but still reserve the higher x bit(depends on parameter), the data in these bits will also available to write/read.

5.16.3.2 InBox Description

Inbox has one 1-depth FIFOs for each outbox. Before source writes into it, source should set the target id register and lock the In Box. After the writing finished, source should release the Inbox.

- If Inbox is not empty, it will note the Router to get mail from it. Router will get one mail one time.
The mail in inbox consists of target ID & mail content.
- If one mail is gotten by router, In Box could send an interrupt to Source.
The mail in InBox could be read by other processor via the debug address. However, the read point will not be changed.

Figure 5-53 Out Box diagram



Mails are saved in sram, each outbox has configurable depth FIFO(1~256), all FIFO share one sram.

If Out Box is not full, it will ack the inbox's req immediately, and router will send mail to it. Router will send one mail one time, if multiple mails need to be send, inbox0 has the highest priority.

The mail in OutBox's FIFO could be read not follow FIFO order via the debug mode pointer. However, the FIFO read point will not be changed.

SRAM control logic arbitrates which channel should read/write SRAM.

Write operation has higher priority than read, and the access order of different channel are: box0 > box1 > box2 > box3 > box4 > box5 > box6 > box7.

Mailbox support debug mode to read mails in the FIFO .Debug pointer are used to access any mail you want regardless of the FIFO order and will not affect the normal read/write pointer.

InBox will store the last mail send to each OutBox.

OutBox will store the former mail in FIFO (include mail ID and content) read by APB, the number of debug mail is parameter configurable in range 1 to 256.

While reading debug mail, address out of box/FIFO range should not been read, or it will get data in other FIFO . Since the mail ID in outbox FIFO are 8bits, so mailbox will combine 4 mail's ID as 1 group and send them together, if (depth of FIFO)%4 $\neq 0$, some higher bits of debug ID are invalid, those bits will be "0".

5.16.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.16.5 User Lock

5.16.5.1 INBOX

Each source processor is able to access all inbox, so an user lock bit is used to arbitration the access authority.

When one processor wants to send mails to other processors, it should find an available inbox to write mail. Processor will read address 0x020 of each box, if the user lock bit is 0, it will set this bit to 1, and use this inbox to send mail; if the user lock bit is 1, indicates that this inbox had been lock by another processor.

After the processor sends all its mail, it should release the inbox. Processor could release the locked inbox by write 0x5a5a5a to address 0x020, and the user lock bit will be clear to 0. Then the inbox could be use by other processors.

But in past project, we often don't use this function, just fix each processor to a inbox, then the working flow of mailbox will be simplified.

5.16.5.2 OUTBOX

Outbox also support this function, but it also not been used mostly.

5.16.6 Sts_busy

sts_busy port is used for low power usage (gated clock). sts_busy is high-active;

- "1" stands for mailbox working, pclk must be active
- "0" stands for mailbox not working, pclk can be gated

5.16.7 Priority Configurable

Register `PRIORITY_LVL_LOW` and `PRIORITY_LVL_HIGH` are used for priority configuration. Every 4bit indicates the inbox id of certain priority. `PRIORITY_LVL_HIGH` has high priority than `PRIORITY_LVL_LOW`. Higher bit of the register indicate higher priority.

Duplicate inbox id should not exist in different priority.

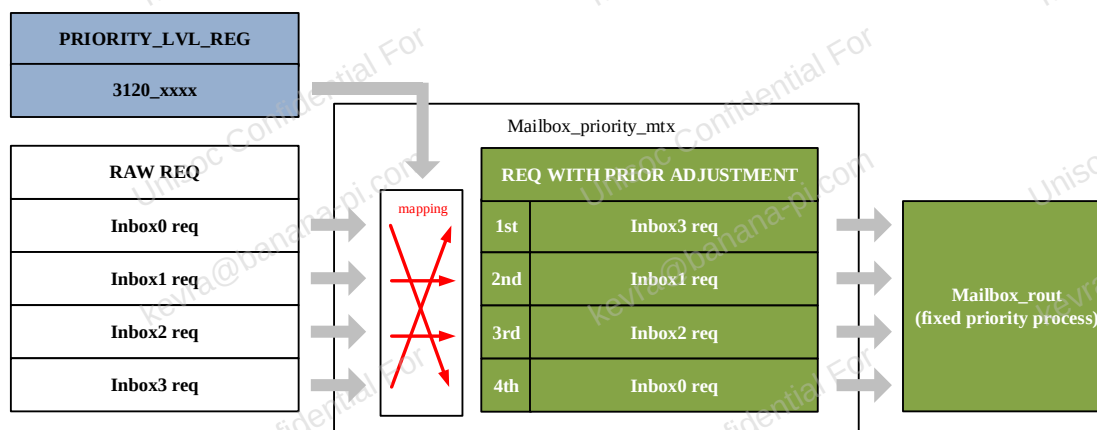
0x20004			PRIORITY_LVL_HIGH (reset 0x0123_4567)													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Priority	1st				2nd				3rd				4th			
Type	RW															
Reset	depends on parameter configure default value															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

0x20004		PRIORITY_LVL_HIGH (reset 0x0123_4567)		
Priority	5th	6th	7th	8th
Type	RW			
Reset	depends on parameter configure default value			

If box number is small than 8, then higher bit of PRIORITY_LVL_HIGH should be configured. For example, only 2 boxes are used, you should configure PRIORITY_LVL_HIGH[31:24] = 8'h01 or 8'h10.

The default value of these register is parameterized. The default parameter value of priority from high to low is 0 > 1 > 2 > 3 > ... c > d > e > f. Thus even the used box number may smaller than maximum box number, the function can work without configuration particularly.

Figure 5-54 Priority adjustment diagram



5.16.8 IP Work Flow Description

There are two different working flow of mailbox: FIFO mode, irq mode.

5.16.8.1 FIFO mode

1. Get inbox: Source Processor could find a free inbox and lock the inbox(if don't use user lock, tie processors with specific in/outbox).
2. Write mail: Source Processor initial one mail and write it into inbox, then write trigger for each mail.
3. Route: In Box should note the Router to get the mails from it. If Router gets one mail(or the target outbox was full/overflow), it will initial an interrupt to Source Processor.
4. Send mail to outbox: Router gets the mail and sends it to the Outbox of target processor. If the Outbox is full. It will block inbox's request and wait for not full.
5. Outbox IRQ: If the Outbox is not empty/empty/full, it will init one interrupt to mention target processor. Target processor could read mail/fifo_status one by one.
6. After all mails are written, Source Processor unlocks the in box(if locked).

5.16.8.2 Irq mode

1. Get inbox: Source Processor could find a free inbox and lock the inbox(if don't use user lock, tie processors with specific in/outbox).
2. Write ID: Source Processor do not write mails content , only write target ID into inbox, then write trigger for each mail.
3. Route: In Box should note the router to get the irq from it, there are no mails need to transfer, it will initial an interrupt to Source Processor.
4. Send irq to outbox: When outbox receive the req of inbox's irq, it will send a irq to the target processor. Target processor will access regfile SCR_FIFO_WR_IRQ. of outbox to find out which source send the irq. Since the irq mode didn't use the FIFO, there are no overflow or full problem with this mode.
5. After all irq are written, Source Processor unlocks the in box(if locked).

5.16.9 Application Notes

5.16.9.1 Send mail flow change

Because the bus may not working in strong order flow, if the outbox FIFO is almost full, and the source processor send a burst to it, mailbox will face data overflow risk (FIFO full but some mails still on the way to mailbox).

To avoid this risk, waiting for "send irq" before sending next mail are recommended working flow.

But "send irq" flow will pull down the bus performance, so in rtl0.9, we add a "receive flag" in inbox regfile to reduce the overflow risk. Before sending each mail, software should polling this flag and make sure it's "0".

5.16.9.2 Switch between FIFO/irq mode

Outbox FIFO support FIFO mode & irq mode for different usage, but do not support mode switch except IP initialize stage, or mailbox can't guarantee the correctness of mail.

5.16.9.3 FIFO reset

FIFO reset function has been updated, during FIFO reset, we not only reset the IRQ flags & wr/rd pointers, but also reset the sram of specific outbox. So the FIFO reset will last dozens(may be hundreds, depends on FIFO depth) of cycles.

During this time all other box can't write sram, so all mail written into inbox can't send to outbox, and since the target FIFO is not full, the block IRQ(inbox) will not be active, if source processor still sending mail, will leading to an OVERFLOW IRQ. So polling receive flag before write mail are highly recommend.

Reset flow:

1. Write 1 to reset rf bit(0x0010,bit[0]);
2. Fifo_reset will stay 1 while resetting, keep polling until this bit return to 0;
3. Then the reset is done.
4. FIFO reset will not effect read mail flow but the resetting outbox is not allow to read.
5. After the fifo_reset ends, mailbox will recover its capability of sending/receiving mails automatically, and all mails blocked in inbox will be send to its target immediately.

5.16.9.4 Debug mode

Mailbox allow to read every mail in FIFO through debug mode regardless of FIFO order(offset addr80~bc for inbox mail, 40~13c for outbox ID, 200~9fc for outbox mail). Since the depth of each FIFO are difference and can configure by register, we still reserve all address for compatibility of the max depth(256).

So software should not read the address out of your depth range, or it will get wrong data.

5.16.9.5 FIFO address overflow

Add `fifo_addr_overflow` flag in `regfile(0x0018 bit[31]`, both in/outbox), when the total FIFO depth set by `regfile` are larger than SRAM depth(set by parameter “`FIFO_TOTAL_DEPTH`”), this flag set to “1”, then the setting of FIFO depth should be checked, “0” indicate no error.

5.17 PWM

5.17.1 Overview

Pulse Width Modulation (PWM) is used to control a ringer or buzzer or LCD lights. The PWM can output 0% ~ 100% duration waves for kinds of applications.

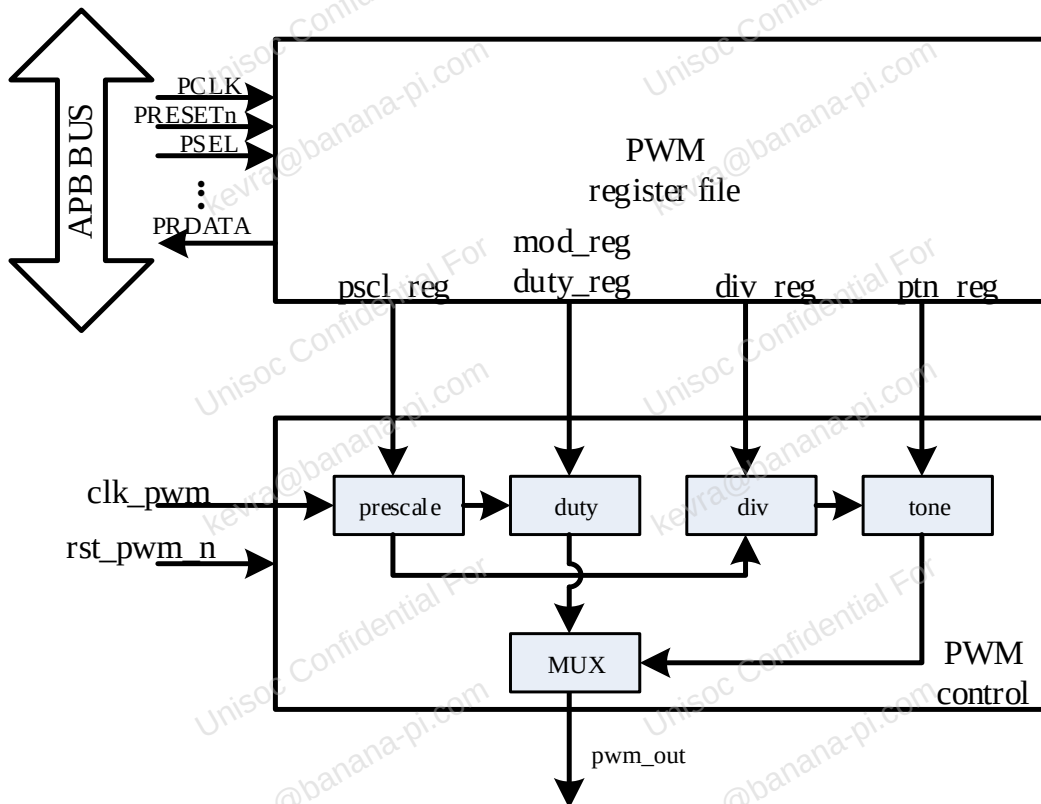
5.17.2 Features

- Support 16bit APB bus register file accessing.
- Support clock pre-scaling, from 1 to 256.
- Support configurable duty ratio, it can cover 0% ~ 100%.
- Duty cycle configuration can be modified dynamically.
- The minimum step for duty cycle adjustment is 1/1024.
- Support PWM output delay.

5.17.3 Function Description

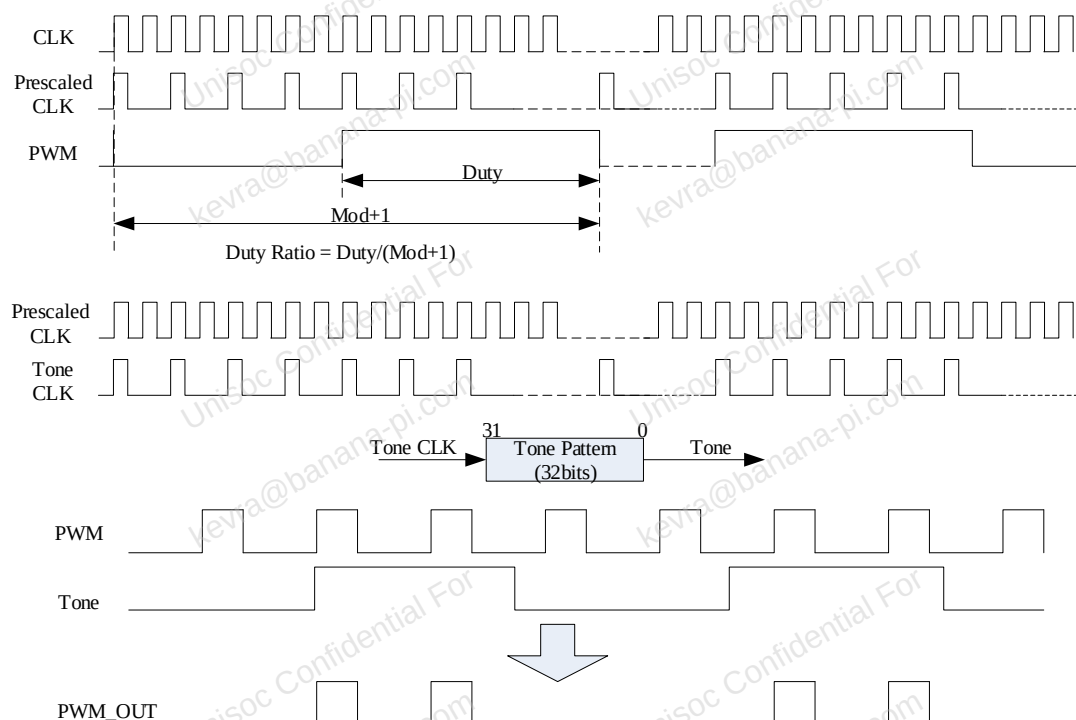
- A PWM functional block diagram is shown below.

Figure 5-55 PWM Block Diagram



- A PWM timing diagram is shown below.

Figure 5-56 PWM Timing Diagram

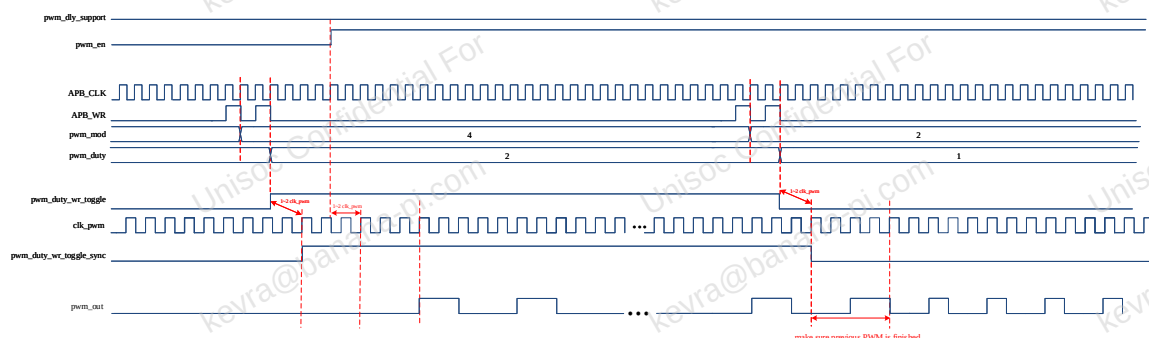


If pre-scale register is P, and mod counter register is M, and duty cycle register is D, then output period T_O is $T_{CLK} \times (P + 1) \times (M + 1)$, and active output period T_{ACT} is $T_{CLK} \times (P + 1) \times D$, and duty ratio is $D / (M + 1)$.

5.17.3.1 PWM_DLY_SUPPORT is 0

- When PWM_DLY isn't supported, PWM first trigger will happen when PWM_EN is from 0 to 1.
- When PWM_EN is valid, if it detects PWM_DUTY write enable, and previous PWM pulse is finished, it will switch to the new configuration.

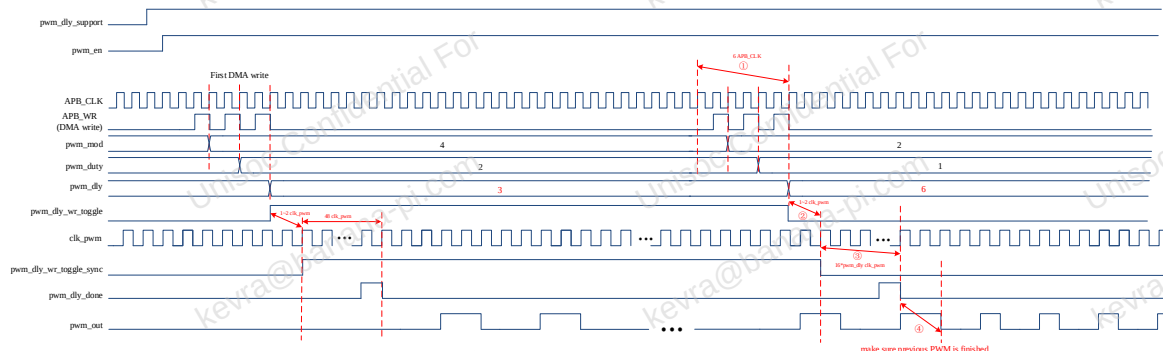
Figure 5-57 PWM delay not support



5.17.3.2 PWM_DLY_SUPPORT is 1

PWM will support delay output. As is shown in figure below, when PWM detects PWM_DLY write enable, it will trigger PWM_DLY timer working. After delay timer is done, and previous PWM pulse is finished, it will switch to the new configuration.

Figure 5-58 PWM delay support



5.17.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.17.5 Application Notes

Software should first select whether PWM delay work mode is supported or not, by setting “PWM_DLY_SUPPORT” of PWM_EN.

5.17.5.1 PWM_DLY_SUPPORT is 0

Software should configure related control registers before “PWM_EN” is set to 1. When all configurations are ready, set “PWM_EN” to 1 to start the PWM output.

PWM_PRESCALE, PWM_MOD and PWM_DUTY can be configured dynamically, where PWM_PRESCALE/PWM_MOD must be configured before PWM_DUTY.

Operation continues until the software writes “PWM_EN” to 0. Once stopped, the PWM output is held low until the next start.

5.17.5.2 PWM_DLY_SUPPORT is 1

Software should set “PWM_EN” to 1 to enable PWM module. After PWM_EN is 1, software should configure PWM_PRESCALE_DMA/PWM_MOD_DMA/PWM_DUTY_DMA/PWM_DLY. When PWM detects PWM_DLY write enable, it will trigger PWM_DLY timer working. After delay timer is done, and previous PWM pulse is finished, it will switch to new configuration.

PWM_PRESCALE_DMA, PWM_MOD_DMA, PWM_DUTY_DMA and PWM_DLY can be configured dynamically, where PWM_PRESCALE_DMA/PWM_MOD_DMA/PWM_DUTY_DMA must be configured before PWM_DLY.

Operation continues until the software writes “PWM_EN” to 0. Once stopped, the PWM output is held low until the next start.

5.18 SCC

5.18.1 Overview

Silicon Characterization and Calibration (SCC) leverage on AVS logic design. This feature combines both pre-production silicon check out and also the post production feature to give more advanced process control and also a new method to improve the product performance.

The design of SCC shall be used in production to gather the silicon speed information.

The design of AVS is in charge of tuning voltage of the CPU core. It uses the report of SCC to adaptively tune the voltage through ADI bus.

5.18.2 Features

SCC design supports the following features:

- Support Ring Oscillators catering for different cell types.
- Support automatic channel connection test.
- Support Gross Ring mode to gauge for silicon data variation.
- Support Aging mode for product feedback and on chip calibration feature.

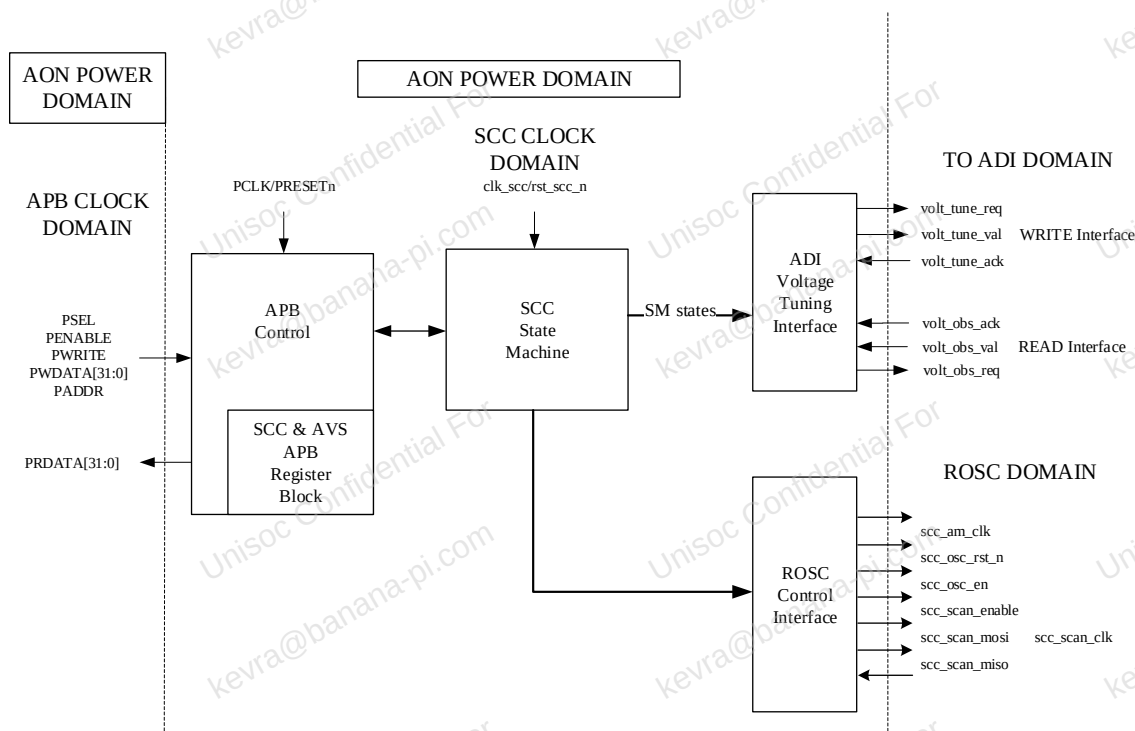
AVS design supports the following features:

- Support Arm® AMBA® APB bus Read and Write monitor.
- Support two selectable ADI channels for voltage observation and tuning.
- Support software configuration of voltage tuning section.
- Support tuning overflow interrupt, raw/mask status and interrupt enable.
- Support tune done interrupt and tune done enable.
- Support limited step for voltage tuning, default max is 8'hff, default min is 8'b0.

5.18.3 Function Description

5.18.3.1 SCC TopFunction Description

Figure 5-59 SCC Logic Block Diagram



SCC extends the AVS controller state machine and the register space to streamline the feature to make the tests development on both the tester and software in a unity view. The design extends the additional interface which controls additional ROSCs as shown in the above logic block diagram. The ROSC Control interface allows a couple of programming i.e. select individual ROSC, a sequence of ROSC and all chains of sequence. This serves as plenty of data points to determine the speed of the silicon as compared to previous product.

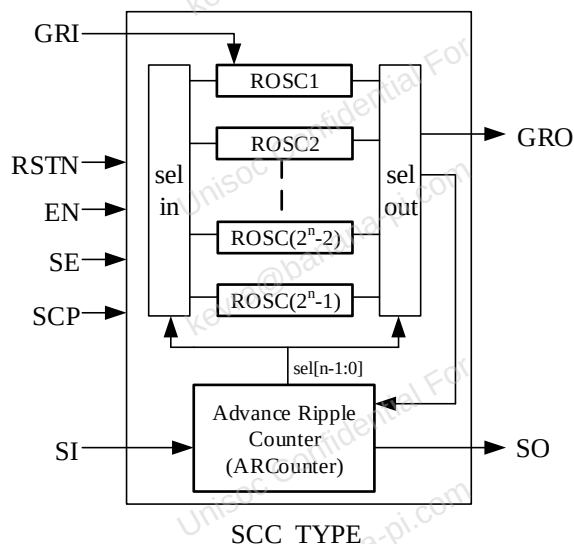
5.18.3.2 SCC Type Function Description

Table 5-8 SCC signals description

Signal Name	bit	Direction	Description
scc_type interface	SCP	1	input
	RSTN	1	input
	SE	1	input
	SI	1	input
	GRI	1	input
	GRO	1	output
	SO	1	output

Signal Name		bit	Direction	Description
	EN	1	Input	SCC EN, high active

Figure 5-60 SCC_TYPE Architecture



This is the Advanced Ripple counters design which allows the usage of the same counter without the need to introduce additional registers to perform capture and load into a parallel registers. The advanced ripple counters will be able to operate at much high frequency as it is packed as a hard macro(scc_type). The serial link which scan the contents of the registers is very scalable. If the latency to obtain the counter values matters, the counters register can be pulled directly as a parallel bus.

5.18.3.3 AVS General Function

AVS is a real-time, continuous, closed-loop power management technology. The AVS technology enables optimum energy management delivery to processors, ASICs, and SOC's by optimizing supply voltages adaptively over process and temperature variations in order to maximize system-level energy savings.

AVS use this report to generate the voltage value for PMIC tuning.

AVS Error Detection

Error would be occurred when voltage tune overflow or underflow. When error, an interrupt should be asserted and states must be stored.

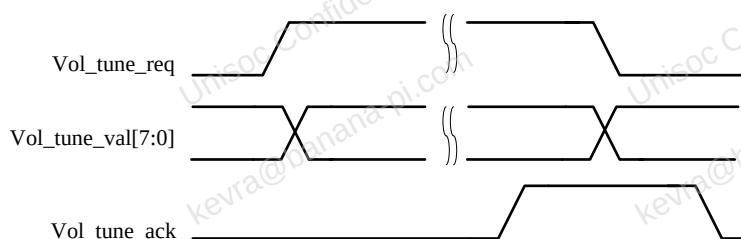
AVS ADI Interface

AVS would synchronize voltage information from ADI, so both read and write operation would be necessary.

For safe information handover, asynchronous handshake method is preferred.

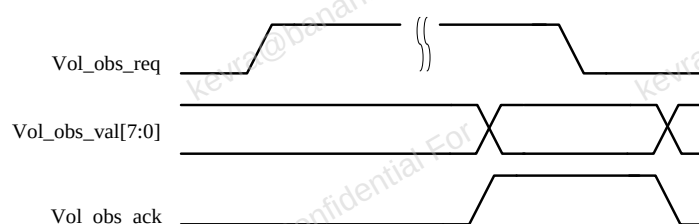
For Write operation in tuning voltage, the following signals should be defined.

Figure 5-61 Write waveform for Handshake



For Read operation: Read handshake has the similar waveform as write, expect that the direction of data bus.

Figure 5-62 Read waveform for Handshake



5.18.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.18.5 Application Notes

Need to ensure a power supply to the power domain

5.19 Spinlock

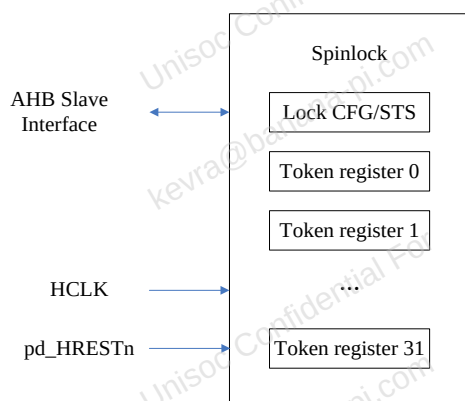
5.19.1 Overview

Hardware spinlocks are provided to assist multiple processors to synchronize access to global resources.

5.19.2 Features

- 32.768kHz RTC-clock triggered
- default 48-bit free-running increasing timer is configured as 64-bit
- 32k (30.52μs) step
- Support single read in register SYST_CNT_SHDW: no need to double read when reading counter value.

Figure 5-63 SpinLock Block Diagram

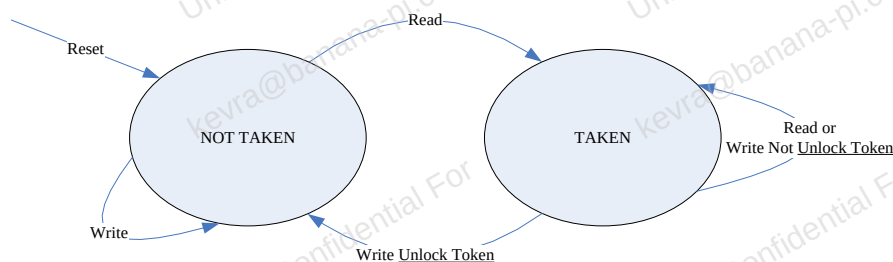


5.19.3 Function Description

5.19.3.1 Take and Release Spinlock

- The spinlock has two states: **Taken** and **Not Taken**.
- After reset, the state of lock is **Not Taken**, equal to 0x0000_0000.
- When the lock is in **Not Taken** state, all write takes no effect.
- When the lock is in **Not Taken** state, a read returns 0x0000_0000, and change to **Taken** state.
- When the lock is in **Taken** state, all write except the Unlock Token(0x55AA10C5) takes no effect. And a read always returns 0x0000_0001.
- A write of the Unlock Token sets the lock back to **Not Taken** state.

Figure 5-64 FSM Diagram



NOTE

There is not support to ensure that a lock is locked and unlocked by the same master by default. This must be ensured in software.

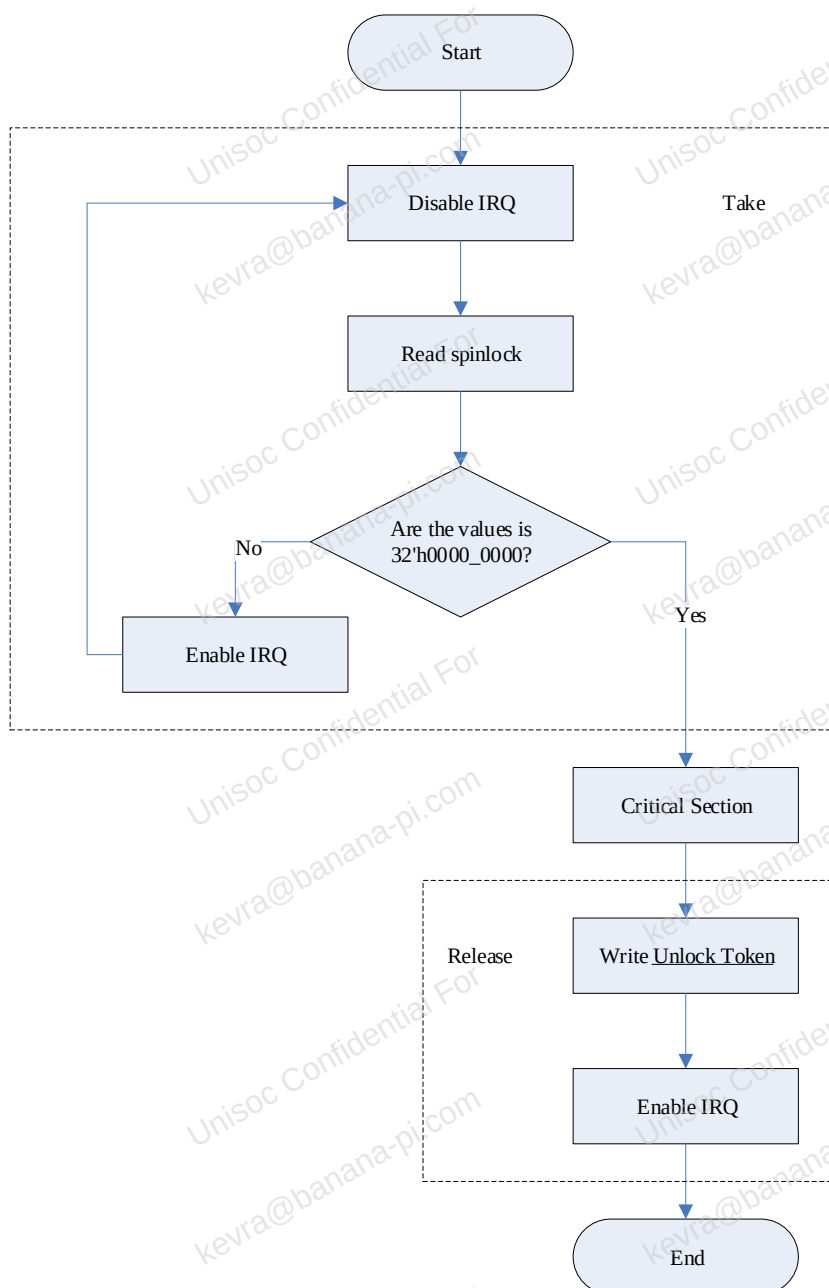
5.19.3.2 Programming Guide

A spinlock should only be held with interrupts disabled. So, before attempting to obtain the spinlock, software should disable interrupts.

Then it should read spinlock. Reading back 0x0000_0000 indicates success. Now it should process the critical section then write Unlock Token(0x55AA10C5) unlock and re-enable interrupts. If the acquisition attempt fails, the acquisition should be reattempted.

To prevent unknown interrupt disabled time, interrupts should be re-enabled and then disabled before reattempting to acquire the lock.

Figure 5-65 Program Flow Diagram



5.19.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.20 SYS_FRT

5.20.1 Overview

System free run timer is a 48-bit (can be parameterized to 33~64bit, default is 48-bit) free-running increasing timer with 32.768kHz RTC clock (30.52μs step). It locates in Always-On Domain (AON), and can be read by all processors. It is used for synchronization between different Co-processor and communication system.

5.20.2 Features

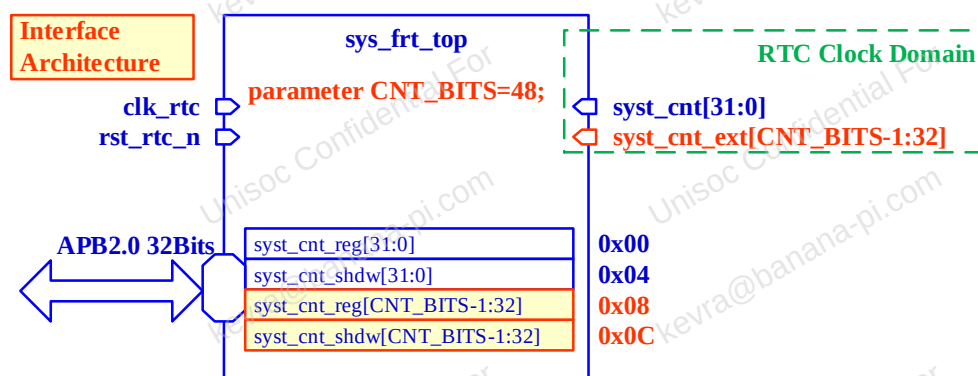
- 32.768kHz RTC-clock triggered
- default 48-bit free-running increasing timer is configured as 64-bit
- 32k (30.52μs) step
- Support single read in register SYST_CNT_SHDW: no need to double read when reading counter value.

5.20.3 Function Description

After hardware reset, `syst_cnt` is 0. And after being enabled, `syst_cnt` runs from 0 to the maximum, $2^{\text{CNT_BITS}}-1$. If `syst_cnt` gets to the maximum, it returns to 0 again and then continues increasing. If no hardware reset occurs, `syst_cnt` will be always free running.

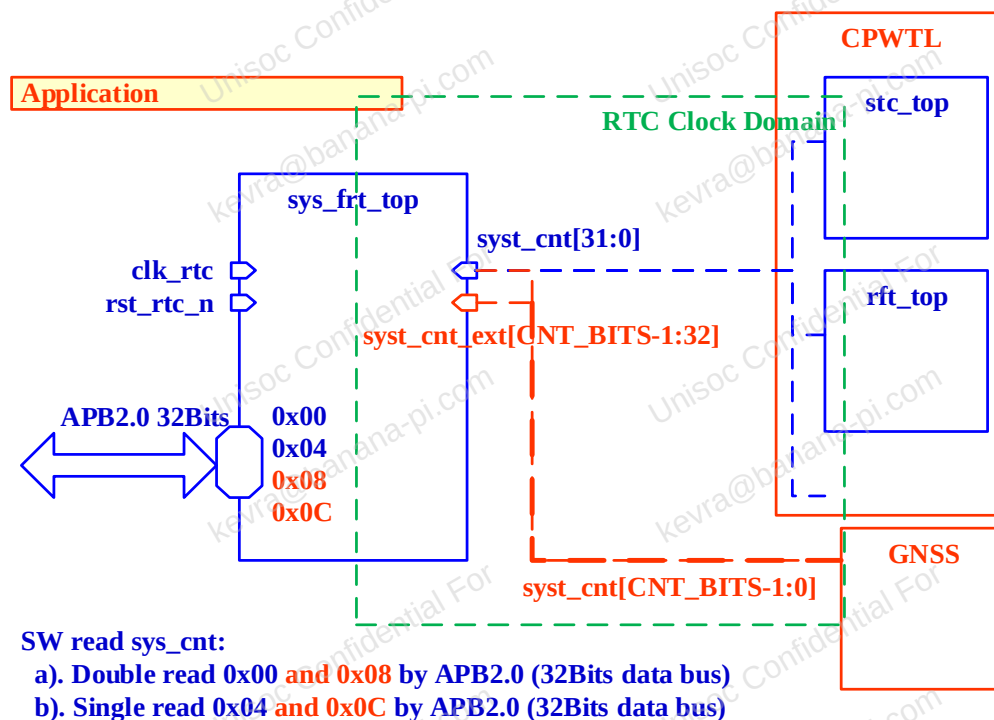
- The bit-width of `syst_cnt` could be 33~ 64 bit which depended on parameter ‘CNT_BITS’. Default value of ‘CNT_BITS’ is 48, and could be re-defined to 33 ~ 64 at SoC integration.
- The `syst_cnt` is counted by 32.768kHz RTC clock, that is, the timer increases by 1 per 30.52μs.
- The timer value can be read by software. If software read SYST_CNT through APB, double-reading method must be used because of different clock domains. If software read SYST_CNT_SHDW, only one read is enough. At the same time, the value can be captured by dedicated capture circuit.

Figure 5-66 Interface Architecture Diagram



Notes: Parameter ‘CNT_BITS’ should be in range 33 ~ 64!

Figure 5-67 Application Diagram



5.20.4 Control Registers

See UIS7885 AON System Control Registers Instructions.

5.20.5 Application Notes

When software begins to use system timer, it should set aon_sys_eb bit to open its PCLK. Its clk_rtc may also be need to opened if necessary

IMPORTANT:

- If software read SYST_CNT through APB, double-reading method MUST be used to read this register.
- If software want get whole CNT_BITS syst_cnt value ({SYS_CNT_EXT, SYS_CNT}), should be double read 0x00 + 0x08.
- If software read SYST_CNT_SHDW or SYS_CNT_SHDW_EXT through APB, only one read can get accurate counter value.

5.21 SYST

5.21.1 Overview

System timer is a 32-bit free-running increasing timer with 1ms step.

The module includes a RTC-clock triggered free-running counter and should be connected on APB.

5.21.2 Features

- RTC-clock triggered
- 32-bit free-running increasing timer
- 1ms step
- Support one configurable alarm, and alarm interrupt generating
- Alarm interrupt can be used as wakeup source during deep sleep
- Support single read: no need to double read when reading counter value

5.21.3 Function Description

After hardware reset, system timer is 0. And after being enabled, system timer runs from 0 to the maximum, $2^{32}-1$. If system timer gets to the maximum, it returns to 0 again and then continues increasing. If no new hardware reset occurs and system timer is kept enabled, nothing can stop system timer.

The counting step is 1ms, that is, the timer increases by 1 per 1ms.

The timer value can be read by software. If software read SYST_CNT, double-reading method must be used because of different clock domains. If software read SYST_CNT_SHDW, only one read is enough. SYST_CNT_SHDW is the shadow of SYST_VALUE by instanced sig_sync_shaw.

Based on system timer free running, it provides an alarm function. When system timer value is equal to alarm value, an interrupt is issued, and this interrupt can be used for system wakeup source during deep sleep.

The interrupt issued by combination logic, this may cause some glitch. It should be noticed when other module use this interrupt, especially edge triggered.

5.21.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.21.5 Application Notes

Including module limitation, mode description, RAM input and output formats, special requirement, software control flow, description, etc.

- When software begins to use system timer, it should set sys_eb bit and sys_rtc_eb bit in global control register.
- When software completes using system timer, it should disable sys_eb bit and sys_rtc_eb bit in global control registers.
- When software uses alarm function, it should configure sys_alm and sys_irq_en.

After receiving interrupt issued by a system timer, software can check interrupt status by reading sys_irq_mask, and can clear interrupt by writing 1 to sys_irq_clr.

IMPORTANT: If software read SYST_CNT, double-reading method MUST be used to read this register. If software read SYST_CNT_SHDW, only one read can get accurate counter value.

5.22 THM_TOP

5.22.1 Overview

A thermal sensor controller that can handle the sensor working configuration, temperature detect flow, simply data processing, interrupt management functions. Controller work in 32k RTC clock.

Analog part of thermal sensor support frequency up to 250kHz.

5.22.2 Features

- Configurable hot/overheat temperature monitor and send interrupt
- Configurable overheat auto alarm logic
- Temperature read by software
- Configurable interrupt mode
- Configurable monitor temperature mode. (1 or 2, 4, 8, 16 sampling average, default 1 sample)
- Configurable temperature sample interval time.
- Sensor controller parameter such as temperature threshold, monitor parameter can change at any time. After changes the configuration, Sensor controller should start to work in new parameter.
- Each sensor's interrupt even can enable or disable individually.
- Each sensor's enable can enable or disable individually.
- Overheat even can cause an interrupt or directly send overheat alarm to top level ASIC logic.
- Sensor controller can access more than eight sensor.

5.22.3 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.22.4 Application Notes

The software read/write registers through APB Bus.

The THM works in 32kHz/250kHz clock domain. The software read/write registers through APB Bus.

Configure steps:

1. Set sensor work parameter.
2. Read `thm_set_ready_st`; if it is 0, set `rf_thm_set_ready` and `rf_thm_mon_en` to 1.
3. Set `rf_thm_en` to 1.
4. Set test sensor's `rf_sensor_en` to 1.

Then THM sensor controller will load the setting of sensor and begin to detect the temperature under setting.

Use `rf_thm_soft_reset` to execute soft reset steps:

1. Set `rf_soft_reset` to 1 to active soft reset.
2. Read THM_CTL register, check `thm_soft_reset_thm_st` and `thm_soft_reset_overheat_st` are both 1.
3. Set `rf_soft_reset` to 0 to invalid soft reset.
4. Read THM_CTL register, check `thm_soft_reset_thm_st` and `thm_soft_reset_overheat_st` are both 0.

When use `rf_thm_overheat_alarm_clr` to clear `thm_overheat_alarm_raw_sts`:

1. Read thm_overheat_alarm_clr_st, check it is 0
2. Write rf_thm_overheat_alarm_clr to 1.

The default initial value of the timing control counters in the module (32k clk):

- Det_peri_cnt 20'h800 0.064 second gap between two det_even_plus
- Sensor_mon_cnt 16'h100 0.008 second gap between two sample

The time is calculated with formula.

Period = detect_period[31:0] × clock_period

THM's most control register will reload configuration after THM_CTL.rf_thm_set_rdy pulse coming, but THM_CTL.rf_thm_en could disable the whole module once it set to disable.

While setting the thresholds of thm, these rules should be follow: Toverheat > Thot ,If the threshold temperature didn't follow the rule, the judgment of FSM state will be wrong.

In THM, rf_thm_det_period is far more than sensor_mon_cnt, in order that THM run normally. But if set sensor_det_peri_load to a small number, THM timing may be wrong. The sensor control signal thm_run become uncertainty.

- When rf_thm_sel_convert is set to 1:
rf_thm_det_period is greater than $(61 \times \text{sample_times} + (\text{sample_times} - 1) \times \text{rf_thm_mon_Period}) \times \text{SENSOR_NUM}$. Sample_times = $2^{\text{rf_thm_mon_mode}} + \text{rf_thm_max_min_en} \times 2$.
- When rf_thm_sel_convert is set to 0:
rf_thm_det_period is greater than $(45 \times \text{sample_times} + (\text{sample_times} - 1) \times \text{rf_thm_mon_Period}) \times \text{SENSOR_NUM}$. Sample_times = $2^{\text{rf_thm_mon_mode}} + \text{rf_thm_max_min_en} \times 2$.

SENSOR_NUM is one controller control sensor's number. When set rf_thm_set_rfy = 1 to the first valid temperature time is $(\text{rf_thm_det_period} + ((45/61) \times \text{sample_times} + (\text{sample_times} - 1) \times \text{rf_thm_mon_period}) \times \text{SENSOR_NUM}) \times \text{clk_period}$. Later every update temperature time is rf_thm_det_period × clk_period.

5.23 Tmr_async_top

5.23.1 Overview

The module is a general-purpose asynchronous timer.

5.23.2 Features

- 64-bit/32-bit decreasing counter
- Support one-time mode and period mode
- Support configurable counting value
- Can be used as wakeup source during deep sleep
- Support single read: no need to double read when reading counter value
- Support multiple load: no need to check busy bit before load tmr
- Support read counter value immediately after loading tmr: once tmr is loaded, the counter value is updated immediately

5.23.3 Function Description

Tmr_async_top is a general-purpose timer. Timer frequency generally is 32.768kHz RTC clock, or other clock such as 26MHz. The frequency determines its precision, if it's 32.768kHz, the precision is 30.52μs. If is 26MHz, the precision is 38.461ns.

5.23.3.1 The 64 bit timer configuration method

In 64 bit mode, the load value process and read value process has to be divided by 2 steps.

During the load value process, there are 2 ways to load data into the timer.

Firstly, if the timer is running, once the LOAD_HI or LOAD_LO has been updated, the newly value will be sent to the timer as soon as possible. And also because the timer is running, so the value will only been updated by the HI or LO, the other part will still be the running one. And attention please, loading value when the timer is running can only be used in the loading idle state which means no other value is being loaded at the same time. For 32 bit mode, the load method will also be changed to match the behavior in 64 bit mode.

Secondly, when the timer is stop, the software can casually change the LOAD_HI or LOAD_LO. Once the software make the timer running again, the {HI,LO} will be sent to the timer and the timer will decrease from {HI,LO}. Also during the timer is stop, if the software doesn't update the LOAD_HI or LOAD_LO. The timer will still decrease from the old {HI,LO} which was written long before.

5.23.3.2 Each asynchronous Timers

Each timer has a 64-bit or 32-bit decreasing counter. This kind of timer provides two operating modes: one-time mode and period mode.

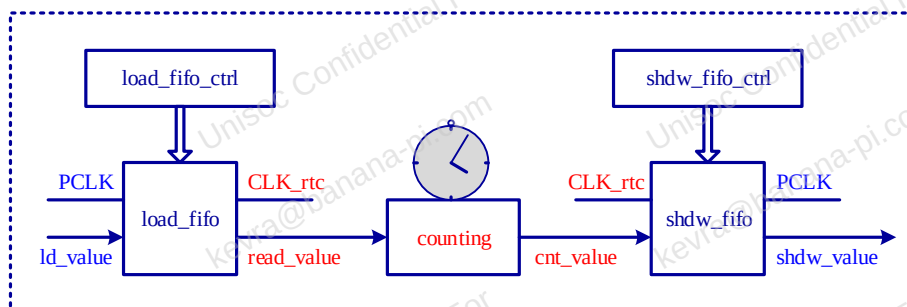
If the timer is configured in one-time mode, counter decreases from setting value to 0. When counter gets to 0, an interrupt is issued. This interrupt is hold until software clears it by writing corresponding clear bit. The counter also stays at 0 until software load it again.

If the timer is configured in period mode, counter decreases from setting value to 0. When counter gets to 0, an interrupt is issued. Then counter goes back to setting value again, and continues to decrease from setting value. The interrupt will hold until software clears it by writing corresponding clear bit.

If triggering clock is RTC clock, these timers can be used to wake up system during deep sleep.

5.23.3.3 Architecture

Figure 5-68 timer architecture



5.23.4 Control Registers

See UIS7885 AON System Control Registers Instructions.

5.23.5 Application Notes

When software uses RTC clock triggered timer, it should set `tmr_eb` bit and `tmr_rtc_eb` bit in global control register.

Before opening `tmr_run`, software should configure all control registers: `timer_load_lo`, `timer_load_hi`, `timer_width_sel`, `timer_mode`, `timer_int_en`. And for the safe use, software can trigger the `timer_int_clr` to clear the interruption, which may still save in the `timer_int` register.

Then `tmr_run` is set to 1 to open corresponding timer.

After receiving interrupt issued by a timer, software can check interrupt status by reading `timer_int_mask_sts`, and can clear interrupt by writing 1 to `timer_int_clr`.

When software completes using RTC clock triggered timer, it should disable `tmr_run` firstly, then it disables `tmr_rtc_eb` and `tmr_eb` in global control registers.

It's IMPORTANT that there are 3~5 RTC clock cycles delay from setting `timer_load_lo` or `timer_load_hi` to counter setting success. that is, if `N` is configured in `timer_load_lo`, the real counting number is `N + 3` or `N + 4` or `N + 5`.

- If the timer is configured in one-time mode, counter decreases from setting value to 0. When counter gets to 0, an interrupt is issued. The counter stays at 0 until software load it again.
- If the timer is configured in period mode, counter decreases from setting value to 0. When counter gets to 0, an interrupt is issued, and counter is reloaded by setting value, and then counter continues to decrease from setting value.

There are two ways to load value to the timer.

- The first one is mainly used. It shall stop the timer write new value to `timer_load_lo` & `timer_load_hi`. After finish refresh the timer load register, software shall set `tmr_run`=1 and the new value will be sent to the timer at this time.
- The second one is load value to the timer when timer is running. This method is mainly used in debug mode. Compared with the first method, in this situation, timer will refresh the related 32bit when the `timer_load_lo` or `timer_load_hi` register is written. Also this action will spend 3~5 PCLK.

In 64bit mode, if software only refresh one load register, another 32bit value will still be the running one. For example, when the timer is running at value of `0x0000_ffff_0000_aaaa`, and software write `0x0000_aaaa` to `timer_load_high`, the next value will be `0x0000_aaaa_0000_aaaa`. If the value is in processing to the timer, the software read value will be stable at `0x0000_aaaa_0000_aaaa`.

5.24 USB2.0

This document describes the functionality of the built-in USB OTG HS/FS controller. In the following text the UNISOC USB OTG HS/FS controller will be referred to as the OTG Controller or the core.

5.24.1 Features

Below is the summary of all UNISOC OTG HS/FS controller features.

- Fully compliant with USB 2.0 specification
- Support the full set of OTG features including session request protocol (SRP) and host negotiation protocol (HNP), compliant with the OTG specification v1.3
- Operates either as the device or host controller
- All three USB speed modes (HS, FS and LS) supported

- Standard VP/VM USB 1.1 transceiver interface (for FS/LS operation only)
- Support suspend and resume signaling
- Soft connect/disconnect features available
- 16 IN and 16 OUT endpoints (including control EP0)
- Dynamic endpoint allocation to maximize number of devices supported 32KB of FIFO available, dynamic FIFO sizing
- Support DMA access to FIFOs
- High-level Arm® AMBA® AHB-compatible CPU interface
- Hardware transaction scheduler to maximize performance
- Support Link Power Management

5.24.2 Introduction

The OTG controller is a versatile design that provides in a single core:

- The function controller of a high-speed/full-speed USB peripheral.
- Dual-role USB controller for point-to-point On-The-Go (OTG) communications with another USB function (which can be either high-speed, full-speed or low-speed).
- The host controller for a multi-point (when connected to a hub) USB system.

The core complies both with the USB 2.0 standard specification for high-speed and full speed transfers and with the On-The-Go supplement to the USB 2.0 specification. The core also supports split transactions, which in turn allows it the full support of the USB 2.0 hub capabilities.

The OTG controller offers 16 IN and 16 OUT unidirectional endpoints that can be configured as 16 bidirectional fully flexible endpoints (including control endpoint 0). Endpoint 0 is the EP for control transfer, and other endpoint can handle transaction of bulk, isochronous, interrupt. Further, the endpoints can also be allocated to different target device functions on the fly maximizing the number of devices that can be simultaneously supported. Every active endpoint requires a FIFO associated with it.

The core's RAM interface connects to a single synchronous single-port RAM block that is used for all the endpoint FIFOs. The FIFO for endpoint 0 is required to be 64 Bytes deep. The RAM interface is configurable with regard to the other endpoint FIFOs, which may be from 8 to 8 192 Bytes in size and can buffer either 1 or 2 packets. Separate FIFOs may be associated with each endpoint: alternatively a Tx endpoint and the Rx endpoint with the same endpoint number can be configured to use the same FIFO, provided they can never be active at the same time.

The core also includes a 32-bit AHB bus interface, running at a wide range of bus speeds. There is also support for DMA access to the endpoint FIFOs. From the USB physical interface side, the OTG controller provides a UTMI + or ULPI interface for connecting to a USB PHY.

The OTG controller provides all the encoding, decoding and checking needed in sending and receiving USB packets. The CPU is only interrupted when endpoint data has been successfully transferred or DMA transfer complete. When acting as the host, the core additionally maintains a frame counter and automatically schedules SOF, isochronous, interrupt and bulk transfers. It also includes full support for the session request and the host negotiation protocols used in point to-point communications.

The core offers a range of test modes primarily the four test modes for high-speed operation described in the USB 2.0 specification chapter 7.1.20. It also includes options that allow it to be forced into Full-speed mode, High-speed mode or Host mode. The last of these may be useful in helping to debug PHY problems in hardware.

5.24.3 Functional Description

The OTG controller has two main modes of operation: Peripheral and Host modes.

In Peripheral mode, the core encodes, decodes, checks and directs all USB packets sent and received. Control, bulk, isochronous and interrupt transactions are supported. IN transactions are handled through the TX FIFOs, OUT transactions are handled through the RX FIFOs.

In Host mode, the core behavior depends on whether it is linked up for point-to-point communications with another USB device or whether it is attached to a hub. When attached to another USB device, the core offers the range of capabilities needed in order to act as a Host in point-to-point communications with this USB device. When attached to a hub, it provides the facilities required to act as the host to a number of devices, supported simultaneously.

When operating in Host mode and used for point-to-point communications with a single other USB device (which can be high-speed, full-speed or low-speed), the core can support control, bulk, isochronous or interrupt transactions. IN transactions are handled through the Rx FIFOs, OUT transactions are handled through the TX FIFOs. As well as encoding, decoding and checking the USB packets sent and received, the core will also automatically schedule Isochronous endpoints and interrupt endpoints to perform one transaction every n frames/micro frames (or up to three transactions if the high-bandwidth option is selected), where n represents the polling interval that has been programmed for the endpoint. The remaining bus bandwidth is shared equally amongst the control and bulk endpoints.

When attached to a hub, the core continues to offer the above facilities but it further needs to be programmed with details of

- The function address of the target device.
- The operating speed of the target device (so that the appropriate speed conversion can be carried out).

If the target device is a full-speed or low-speed device that is accessed through a high-speed hub, the endpoint additionally needs to be programmed with the function address and port number of the hub.

To maximize the number of devices that can be supported simultaneously in this mode, the core also offers the facility for endpoints to be dynamically reprogrammed on a per transaction or per-transfer basis. This allows, for example, the endpoint 0 controller to be used to access endpoint 0 of any connected device or, if preferred, a Tx and Rx endpoint pair can be set up to handle the control protocol required to support a device's endpoint 0. The device may be required to power the VBus to 5 volts as the A device of the connection (source of power and default host) or, as the B device (default peripheral), to be able to wake the A device by charging the VBus to 2 volts. Outputs from the core indicate when these charging options are required.

Whether the core initially operates in Host mode or in Peripheral mode depends on whether it is being used in an A-device or a B-device, which in turn depends on whether the ID pin of the transceiver is low or floating. When the core is operating as an A-device, it is initially configured to operate in Host mode. When operating as a B-device, the core is initially configured to operate in Peripheral mode. However, a Host Req bit is provided in the OTG device control register ([OTG_DEVCTL](#)) through which the CPU can request that the B-device becomes the host the next time there is no activity on the USB.

Information on whether the core is acting as an A-device or as a B-device and on whether the device it is connected to is high-speed, full-speed or low-speed is also recorded in OTG device control register ([OTG_DEVCTL](#)), along with information about the level of the VBUS relative to the high and low voltage thresholds used to signal session start and session end.

The procedures for session request and for transferring host/peripheral roles between the devices at either end of the connection are described in [OTG session request \(SRP\)](#) and [SHost negotiation \(HNP\)](#) respectively. The transfers that are made are all subject to the standard USB data transfer protocols.

5.24.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.24.5 Operation in Peripheral mode

This chapter describes the details of how the OTG controller handles the peripheral mode tasks.

5.24.5.1 USB reset

When a reset condition is detected on the USB, the core performs the following actions:

1. Set OTG_FADDR to 0
2. Set OTG_INDX to 0
3. Flush all endpoint FIFOs
4. Clear all control/status registers
5. Enable all interrupts, except Suspend
6. Generate a Reset interrupt

5.24.5.2 Peripheral Mode: IN transactions

When the core is operating in Peripheral mode, data for IN transactions is handled through the core's Tx FIFOs. The sizes of the Tx FIFOs for Endpoints 1 to 15 are through the OTG TX FIFO size register. The maximum size of data packet that may be placed in a TX endpoint's FIFO for transmission is programmable and is determined by the value written to the OTG TX max packet size register for that endpoint (maximum payload, number of transactions/microframe (where applicable)). Where dynamic FIFO sizing is selected, the use of single or double packet buffering is part of the specification for the endpoint FIFO.

When double packet buffering is enabled, two data packets can be buffered in the FIFO: when single packet buffering is enabled, only one packet can be buffered even if the packet is less than half the FIFO size. The maximum packet size set for any endpoint must not exceed the FIFO size. You should also note that the OTG_TXMAXP should not be written to while there is data in the FIFO as unexpected results may occur.

5.24.5.3 Single packet buffering

If the size of the Tx endpoint FIFO is less than twice the maximum packet size for this endpoint (as set in the OTG_TXFSZ register), only one packet can be buffered in the FIFO and single packet buffering is enabled. As each packet to be sent is loaded into the Tx FIFO, the TxPktRdy bit in OTG TX control and status (DEVICE) register needs to be set.

If the AutoSet bit in OTG_TXCSR is set, the TxPktRdy bit will be automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, TxPktRdy will always have to be set manually (i.e. by the CPU). When the TxPktRdy bit is set, either manually or automatically, the FIFONotEmpty bit in OTG_TXCSR is also set and the packet is ready to be sent.

When the packet has been successfully sent, both TxPktRdy and FIFONotEmpty will be cleared and the appropriate Tx endpoint interrupt generated (if enabled). The next packet can then be loaded into the FIFO.

5.24.5.4 Double packet buffering

If the size of the TX endpoint's FIFO is at least twice the maximum packet size for this endpoint (as set in the [OTG_TXFSZ](#) register), two packets can be buffered in the FIFO and double packet buffering is enabled. As each packet to be sent is loaded into the TX FIFO, the TxPktRdy bit in OTG_TXCSR needs to be set.

If the AutoSet bit in OTG_TXCSR is set, the TxPktRdy bit will be automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, TxPktRdy will always have to be set manually (i.e. by the CPU). When the TxPktRdy bit is set, either manually or automatically, the FIFONotEmpty bit in [OTG_TXCSR](#) will also be set. TxPktRdy is then immediately cleared (and an interrupt generated, if enabled). A second packet can now be loaded into the TX FIFO and TxPktRdy set again (either manually or automatically if the packet is the maximum size). Both packets are now ready to be sent.

When the first packet has been successfully sent, TxPktRdy will be cleared and the appropriate Tx endpoint interrupt generated (if enabled) to signal that another packet can now be loaded into the Tx FIFO. The state of the FIFONotEmpty bit at this point indicates how many packets may be loaded.

- If the FIFONotEmpty bit is set then there is another packet in the FIFO and only one more packet can be loaded.
- If the FIFONotEmpty bit is clear then there are no packets in the FIFO and two more packets can be loaded.

5.24.5.5 High-bandwidth isochronous/interrupt endpoints

In High-speed mode, Isochronous Tx endpoints can transmit up to three USB packets in any microframe, with a payload of up to 1024 Bytes in each packet, corresponding to a data transfer rate of up to 3072 Bytes per microframe. The core supports this by allowing the user to load data packets of up to 3072 Bytes (i.e. 3×1024 Bytes) into the associated FIFO in a single transaction.

From the point of view of the software in the CPU, the operation is then exactly as described above for Single Packet Buffering or Double Packet Buffering (as appropriate) except that TxPktRdy will always need to be set manually (i.e. by the CPU) as AutoSet does not operate with high-bandwidth Isochronous transfers.

Any data packet loaded into the FIFO that is larger than the maximum payload is automatically split into USB packets of the maximum payload, or smaller, for transmission over the USB. The number of USB packets transmitted per microframe and the maximum payload in each packet is defined through the *OTG_TXMAXP* register. bit[10:0] of the *OTG_TXMAXP* register determine the maximum payload in any USB packet while bit[12:11] determine the maximum number of such packets that can be sent in one microframe (2 or 3). Together, these set the maximum size of packet that can be loaded into the FIFO. At least one USB packet will always be sent: the number of further USB packets sent in the same microframe will depend on the amount of data loaded into the FIFO. The TxPktRdy bit will be cleared and an interrupt generated only when all the packets have been sent.

Each USB packet is sent in response to an IN token. If at the end of a microframe, the core has not received enough IN tokens to send all the USB packets (e.g. because one of the IN tokens received was corrupted), the remaining data will be flushed from the FIFO. The TxPktRdy bit will then be cleared and the IncompTx bit in the *OTG_TXCSR* register set to indicate that not all of the data loaded into the FIFO was sent.

5.24.5.6 Optional special handling

The packets transferred in Bulk operations are defined by the USB Specification to be either 8, 16, 32, 64 or 512 Bytes in size, with the 512 Byte option only applying to High Speed transfers. For some system designs, however, it may be more convenient for the application software to write larger amounts of data to an endpoint in a single operation than can be transferred in a single USB operation. A particular case in point is where the same endpoint is used for high-speed transfers of 512 Bytes under certain circumstances but for full-speed transfers under other circumstances.

When operating at full-speed, the maximum amount of data transferred in a single operation is then just 64 Bytes. To cater for such circumstances, the core includes a configuration option which, if selected, allows larger data packets to be written to Bulk Tx endpoints which are then split into packets of an appropriate (specified) size for transfer across the USB bus. (A similar option exists for reading from Bulk Rx endpoints in larger volumes than individual USB packets.) Under this option, the *OTG_TXMAXP* register for the endpoint is increased to 16 bits and the bottom 11 bits of the register define the payload for each individual transfer, while the top 5 bits define a multiplier. The application software can then write data packets of size multiplier payload to the FIFO which the core will then split into individual packets of the stated payload for transmission over the USB. From the application software's point of view, the resulting operation will not differ from the transmission of a single USB packet except in the size of the packet written. This facility is offered as a configuration option rather than as a standard feature because it increases the gate count.

Note: This feature is only for use with Bulk endpoints and, in accordance with the USB Specification, the payload must be either 8, 16, 32, 64 or 512 Bytes with the 512 Byte option only applicable for High-Speed transfers. The

payload recorded in the [OTG_TXMAXP](#) register must also match the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. The associated FIFO must also be large enough to accommodate the data packet prior to being split.

5.24.5.7 Peripheral mode: OUT transactions

When the core is operating in Peripheral mode, data for OUT transactions is handled through the core's Rx FIFOs. The sizes of the Rx FIFOs for Endpoints 1 to 15 are determined through the [OTG_RXFSZ](#) register. The maximum amount of data received by an Rx endpoint in any frame or microframe (in High-speed mode) is programmable and is determined by the value written to the [OTG_RXMAXP](#) register for that endpoint. (maximum payload × number of transactions/microframe (where applicable)). The use of single or double packet buffering is part of the specification for the endpoint FIFO.

When double packet buffering is enabled, two data packets can be buffered in the FIFO: when single packet buffering is enabled, only one packet can be buffered even if the packet is less than half the FIFO size.

NOTE

The maximum packet size should not exceed the FIFO size.

5.24.5.8 Single packet buffering

If the size of the Rx endpoint FIFO is less than twice the maximum packet size for this endpoint (as set in the [OTG_RXFSZ](#) register), only one data packet can be buffered in the FIFO and single packet buffering is enabled. When a packet is received and placed in the Rx FIFO, the RxPktRdy bit and the FIFOFull bit in the [OTG_RXCSR](#) are set and the appropriate Rx endpoint is generated (if enabled) to signal that a packet can now be unloaded from the FIFO. After the packet has been unloaded, the RxPktRdy bit needs to be cleared in order to allow further packets to be received.

If the AutoClear bit in [OTG_RXCSR](#) is set and a maximum-sized packet is unloaded from the FIFO, the RxPktRdy bit is cleared automatically. The FIFOFull bit is also cleared. For packet sizes less than the maximum, RxPktRdy will always have to be cleared manually (that is, by the CPU).

5.24.5.9 Double packet buffering

If the size of the Rx endpoint FIFO is at least twice the maximum packet size for the endpoint (as set in the [OTG_RXFSZ](#) register), two data packets can be buffered and double packet buffering is enabled. When the first packet to be received is loaded into the Rx FIFO, the RxPktRdy bit in [OTG_RXCSR](#) is set and the appropriate Rx endpoint interrupt is generated (if enabled) to signal that a packet can now be unloaded from the FIFO.

Note: The FIFOFull bit in [OTG_RXCSR](#) is not set at this point: it is only set if a second packet is received and loaded into the Rx FIFO. After the first packet has been unloaded, RxPktRdy needs to be cleared in order to allow further packets to be received.

If the AutoClr bit in [OTG_RXCSR](#) is set and a maximum-sized packet is unloaded from the FIFO, the RxPktRdy bit will be cleared automatically. For packet sizes less than the maximum, RxPktRdy will always have to be cleared manually (by the CPU). If the FIFOFull bit was set to 1 when RxPktRdy is cleared, the core will first clear the FIFOFull bit. It will then set RxPktRdy again to indicate that there is another packet waiting in the FIFO to be unloaded.

5.24.5.10 High-bandwidth isochronous endpoints

In High-speed mode, Isochronous Rx endpoints can receive up to three USB packets in any microframe, with a payload of up to 1024 Bytes in each packet, corresponding to a data transfer rate of up to 3072 Bytes per microframe.

Note: There is not support for high-bandwidth Interrupt transactions in Device mode.

The core supports this by automatically combining all the USB packets received during a microframe into a single packet of up to 3072 Bytes (i.e. 3×1024 Bytes) within the Rx FIFO.

From the point of view of the software in the CPU, the operation is then exactly as described above for Single Packet Buffering or Double Packet Buffering (as appropriate) except that RxPktRdy will always need to be cleared manually (i.e. by the CPU) as AutoClear does not operate with high-bandwidth Isochronous transfers. The maximum number of USB packets that may be received in any microframe and the maximum payload of these packets are defined through the [OTG_RXMAXP](#) register. bit[10:0] of the OTG_RXMAXP register determine the maximum payload in any USB packet while bit[12:11] determine the maximum number of these packets that may be received in a microframe (2 or 3). The number of USB packets sent in any microframe will depend on the amount of data to be transferred, and is indicated through the PIDs used for the individual packets. If the indicated number of packets have not been received by the end of a microframe, the IncompRx bit in the OTG_RXMAXP register will be set to indicate that the data in the FIFO is incomplete. An interrupt will, however, still be generated to allow the data that has been received to be read from the FIFO.

5.24.6 Operation in Host mode

When the Host Mode bit (OTG_DEVCTL register, bit 2) is set, the core operates as a host either for point-to-point communications with another USB device or, when attached to a hub, for communication with a whole range of devices in a multi-point set-up. High-speed, full-speed and low-speed USB functions are supported, both for point-to-point communication and for operation through a hub. (Where necessary, the core automatically carries out the necessary transaction translation needed to allow a low-speed or full-speed device to be used with a USB 2.0 hub.) Control, Bulk, Isochronous or Interrupt transactions are supported. This section looks at the core's actions with regard to Tx endpoints, Rx endpoints, transaction scheduling, entry into/exit from Suspend mode and reset that apply when the core is being used as a host.

5.24.6.1 Host mode: device setup

Prior to accessing any device as a host whether for point-to-point communications or for multi-point communications via a hub the relevant [OTG_TXFAD_n](#) registers need to be set for each used Rx or Tx endpoint to record the function address of the device being accessed. Where a full-speed or low-speed device is connected to the core via a high-speed USB 2.0 hub, details of the hub address and the hub port also need to be recorded in the corresponding [OTG_TXHAD_n](#) and [OTG_TXHP_n](#) registers. This allows the core to support split transactions. In addition the speed at which the device operates (high, full or low) needs to be recorded in the [OTG_TYPE0](#), [OTG_TXTYPE](#) or [OTG_RXTYPE](#) registers for each endpoint that is accessed by the device. For multi-point communications, it should be noted that the settings in these registers record the current allocation of the core's endpoints to the functions associated with the attached devices.

To maximize the number of devices supported, the core allows this allocation to be changed dynamically simply by updating the address and speed information recorded in these registers. Any changes in the allocation of endpoints to device functions need to be made following the completion of any on-going transactions on the endpoints affected. Further information on allocating endpoints to device functions and switching between different allocations is given in the USB core's programmer's guide.

5.24.6.2 Host mode: IN transactions

When the core is operating as a host, IN transactions are handled in a similar manner to the way in which OUT transactions are handled when the core is operating as a peripheral except that the transaction needs first to be initiated by setting the ReqPkt bit in OTG_RXCSR. This indicates to the transaction scheduler that there is an active transaction on this endpoint. The transaction scheduler then sends an IN token to the target function. When the packet is received and placed in the Rx FIFO, the RxPktRdy bit in OTG_RXCSR is set and the appropriate Rx endpoint interrupt is generated (if enabled) to signal that a packet can now be unloaded from the FIFO.

When the packet has been unloaded, RxPktRdy should be cleared. The AutoClear bit in the OTG_RXCSR register can be used to have RxPktRdy automatically cleared when a maximum sized packet has been unloaded from the FIFO. There is also an AutoReq bit in OTG_RXCSR which causes the ReqPkt bit to be automatically set when the

RxPktRdy bit is cleared. The AutoClear and AutoReq bits can be used with an external DMA controller to perform complete Bulk transfers without CPU intervention.

- If the target function responds to a Bulk/Interrupt IN token with a NAK, the core will keep retrying the transaction until any NAK Limit that has been set has been reached.
- If the target function responds with a STALL, however, the core will not retry the transaction but will interrupt the CPU with the RxStall bit in the OTG_RXCSR register set.
- If the target function does not respond to the IN token within the required time (or there was a CRC or bit-stuff error in the packet), the core will retry the transaction. If after three attempts the target function has still not responded, the core will clear the ReqPkt bit and interrupt the CPU with the Error bit in OTG_RXCSR set.

NOTE

In the case of high-bandwidth Interrupt transactions, the host will attempt 2 or 3 transactions during a single microframe and generate an interrupt when all packets have been received. If any of these transactions is not acknowledged by a Device, no further transactions will be attempted during the same microframe.

5.24.6.3 Host mode: OUT transactions

When the core is operating as a host, OUT transactions are handled in a similar manner to the way IN transactions are handled when the core is operating as a peripheral. The TxPktRdy bit in the *OTG_TXCSR* register needs to be set as each packet is loaded into the TxFIFO and the AutoSet bit in *OTG_TXCSR* can be used to cause the TxPktRdy bit to be automatically set when a maximum sized packet has been loaded into the FIFO. Again, the AutoSet bit can be used with an external DMA controller to perform complete Bulk transfers without CPU intervention.

- If the target function responds to the OUT token with a NAK, the core will keep retrying the transaction until any NAK Limit that has been set has been reached.
- If the target function responds with a STALL, however, the core will not retry the transaction but will interrupt the CPU with the RxStall bit in the *OTG_TXCSR* register set.
- If the target function does not respond to the OUT token within the required time (or there was a CRC or bit-stuff error in the packet), the core will retry the transaction. If after three attempts the target function has still not responded, the core will flush the FIFO and interrupt the CPU with the Error bit in *OTG_TXCSR* set.

5.24.6.4 Host mode: transaction scheduling

When operating as a host, the core maintains a frame/microframe counter.

- If the target function is a full-speed or high-speed device, the core will automatically send an SOF/uSOF packet at the start of each frame/microframe.
- If the target function is a low-speed device, a K state will be transmitted on the bus to act as a keep-alive to stop the low-speed device going into Suspend mode.

After the SOF/uSOF packet has been transmitted, the core will cycle through all the configured endpoints looking for active transactions. An active transaction is defined as an Rx endpoint for which the ReqPkt bit is set or a Tx endpoint for which the TxPktRdy bit is set. An active Isochronous or Interrupt transaction will only be started if found on the first transaction scheduler cycle of a frame/ microframe and if the interval counter for that endpoint has counted down to zero. This ensures that only one Interrupt/Isochronous transaction will occur per endpoint per n frames/microframes (or up to three if high-bandwidth support is selected) where n is the interval set via the *OTG_TXINTV* / *OTG_RXINTV* register for that endpoint. An active Bulk transaction will be started immediately, provided there is sufficient time left in the frame/microframe to complete the transaction before the next SOF/uSOF packet is due.

If the transaction needs to be retried (e.g. because a NAK was received or the target function did not respond) then the transaction will not be retried until the transaction scheduler has checked all the other endpoints for active transactions first. This ensures that an endpoint that is sending a lot of NAKs does not block other transactions on

the bus. The core also allows the user to specify a limit to the length of time for NAKs may be received from a particular target before the endpoint is timed out.

5.24.6.5 Babble

The core will not start a transaction until the bus has been inactive for at least the minimum interpacket delay. It will also not start a transaction unless it can be finished before the end of the frame. If the bus is still active at the end of a frame then the core will assume that the function it is connected to has malfunctioned and will suspend all transactions and generate a babble interrupt

5.25 Watchdog

5.25.1 Overview

Watchdog timer is a 32-bit decreasing timer with RTC-clock triggered, and this timer can be used to issue system reset.

5.25.2 Features

- RTC-clock triggered
- 32-bit decreasing timer
- Support 32-bit timer value loading
- Support reset mode for watchdog function
- Support interrupt mode for debug function and timer function
- Support combined mode: first generate interrupt, and then generate reset.
- Support single read: no need to double read when reading counter value
- Support multiple load: no need to check busy bit before load wdg_cnt
- Support read counter value immediately after loading wdg_cnt: once wdg_cnt is loaded, the counter value is updated immediately

5.25.3 Function Description

After hardware reset, watchdog timer is 32'hFFFF_FFFF. And after enabled, watchdog timer decreases from the reset value or from software-loading value.

The counting step is one RTC clock cycle.

- For reset mode, whenever watchdog timer gets to 0, system reset is issued. Then system reboots, and watchdog timer comes back to 32'hFFFF_FFFF, and all controlled registers and status also come back to reset value, except interrupt raw status. Interrupt raw status can be used to judge if or not system rebooting comes from watchdog reset.
- For interrupt mode, an interrupt is issued when watchdog timer gets to wdg_irq_value, that is { wdg_irq_value_high, wdg_irq_value_low}. Then the timer continue to count down until it reach 0. After that, watchdog timer comes back to the loaded value.

Combined mode is the combination of reset mode and interrupt mode. Interrupt will be generated first if gets to WDG_IRQ_VALUE, that is { wdg_irq_value_high, wdg_irq_value_low}. Then software should stop wdg so that wdg reset is not generated. If software cannot response to this interrupt, then wdg reset will be generated finally when wdg_cnt decrease to 0 which reboots the entire system.

After enabled, watchdog timer can be loaded at any time, and software should guarantee the timer is loaded before it decreases to 0.

Reset mode is used as normal watchdog function. And interrupt mode is used as debug mode or as a generic periodic timer.

5.25.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

5.25.5 Application Notes

When software uses watchdog, it should set `wdg_eb` bit and `wdg_rtc_eb` bit in global control register, and also should guarantee `arch_rtc_eb` is set.

Before configuring watchdog, software should write `16'hE551` to `wdg_lock` to unlock watchdog. Then Software configures all control registers: `wdg_ld_value_low`, `wdg_ld_value_high`, `wdg_irq_value_low`, `wdg_irq_value_high`, `wdg_open` and `wdg_new`.

IMPORTANT: `wdg_ld_value_high` should always be configured before writing `wdg_ld_value_low`. Then `wdg_open` is set to 1 to open counter.

Reset mode is used as normal watchdog function. After enabled, watchdog timer can be loaded at any time, and software should guarantee the timer is reloaded before it decreases to 0.

Whenever watchdog timer gets to 0, system reset is issued. Then system reboots, and watchdog timer comes back to `32'hFFFF_FFFF`, and all controlled registers and status also come back to reset value, except interrupt raw status. Interrupt raw status can be used to judge if or not system rebooting comes from watchdog reset.

For interrupt mode, an interrupt is issued when watchdog timer gets to `WDG_IRQ_VALUE`, that is { `wdg_irq_value_high`, `wdg_irq_value_low` }. Then the timer continue to count down until it reach 0. After that, watchdog timer comes back to the loaded value.

After receiving interrupt, software can check interrupt status by reading `wdg_irq_mask`, and can clear interrupt by writing 1 to `wdg_irq_clr`.

Combined mode is the combination of reset mode and interrupt mode. Interrupt will be generated first if gets to `WDG_IRQ_VALUE`, that is { `wdg_irq_value_high`, `wdg_irq_value_low` }. Then software should stop `wdg` so that `wdg` reset is not generated. If software cannot response to this interrupt, then `wdg` reset will be generated finally when `wdg_cnt` decrease to 0 which reboots the entire system.

When software completes using watchdog, it should disable `wdg_open` firstly, then it disables `wdg_rtc_eb` and `wdg_eb` in global control registers.

It's IMPORTANT that:

- Only writing `wdg_ld_value_high` cannot reload watchdog. Both `wdg_ld_value_low` and `wdg_ld_value_high` should be written if watchdog need reloaded. And software must write `wdg_ld_value_high` firstly, then write `wdg_ld_value_low`.
- There are 3-5 RTC clock cycles delay from setting `wdg_ld_value_low` to counter setting success because of different clock domains, just as timer does.
- It's not recommended to read counter value in normal usage, just as timer does.

5.26 CH I3C

5.26.1 Overview

The I3C interface is intended to improve upon the feature of the I2C interface, preserving backward compatibility. Two main concerns are paramount for the I3C interface: The use of as little energy as possible in transporting data and control, while reducing the number of physical pins used by the interface. This specification defines a standard Multi-Drop interface between host processors and peripheral sensors.

5.26.2 Features

- Support SDR protocol.
- Manage address arbitration.
- Support arbitration optimization.
- Support repeat start one time when address NACK using without 7E address mode.
- In-band interrupt (IBI).
- Hot-join request.
- SDR error detection.
- Common Command Code.
- Support DMA mode.
- Dynamic Address Assignment procedure.
- I2C mode (standard and fast speed mode)
- SCL timing option mode (SCL meet 12.6MHz)
- SDA OE delay option (avoid unexpected SDA glitch)

5.26.3 Function Description

5.26.3.1 Hot-Join request

Hot-Join mechanism, to allow slaves to join the I3C bus after it is already configured.

Since the master is not inherently aware when new slaves join the I3C bus, the Hot-Join mechanism provides the following procedure for informing the master about new slaves:

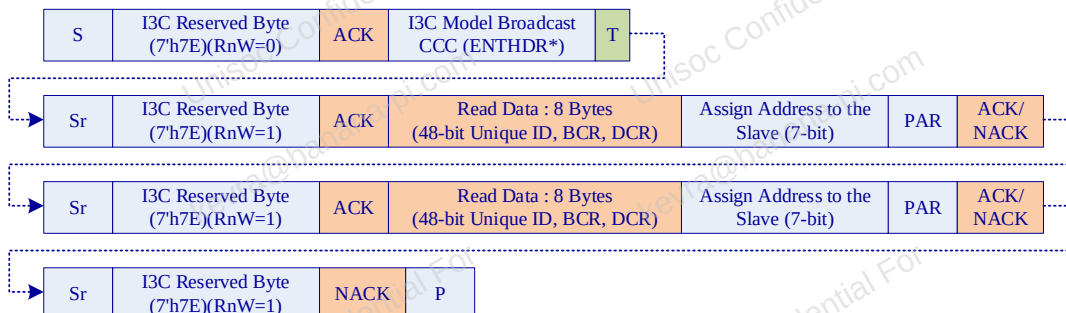
1. The slave shall wait for at least t_{IDLE} of bus available condition.
2. After the slave determines that the bus is idle it may either issue a START, or else wait for a START (not a repeat START).
3. The slave issues the reserved slave address of 7'b0000_010 as an IBI, using W (write) after the START. This requests a Dynamic Address Assignment process.
4. The master shall perform one of 3 actions:
 - NACK the request.
 - ACK the request, but issue a Broadcast CCC to disable Hot-Join by setting the DISHJ bit in the Command Code Enable/Disable Slave Events Command (ENEC/DISEC).
 - ACK the request and then issue a Broadcast Command Code Enter Dynamic Address Assignment (ENTDAA).

Hot-Join

S	7'b0000_010 (RnW=0)	ACK/ NACK	Sr/P
---	---------------------	--------------	------

5.26.3.2 Dynamic Address Assignment

The main master is responsible for performing a Dynamic Address Assignment procedure, in order to provide a unique dynamic address to each device connected to the I3C Bus.



5.26.3.3 In-Band Interrupt (IBI)

In order to request an interrupt, an I3C slave shall wait for the Bus Available Condition and then issue START by pulling the SDA line low. In response, the main master shall start the clock on the SCL line, while leaving the SDA line free at high level. The main master shall process interrupt requests in priority level order. In I3C, priority level controls the order in which slaves' In-Band Interrupt requests and master requests are processed. The priority level of each slave in an I3C bus instantiation is encoded in its slave address, with lower addresses having higher priority.

The master shall do one of the following three things when detect IBI:

- Accept the IBI by providing the ACK bit. If the I3C slave's BCR[2] bit is set to 1, then the master shall read the mandatory data byte that follows the accepted IBI request at any "read" clock speed allowable by the slave. This operation is similar to a "read" from the slave and all the related rules apply. Note that the master cannot avoid receiving the mandatory data byte, since it is transmitted in push-pull mode.
- Refuse the IBI without disabling interrupts. To do this, the master simply passively NACKs to deny the IBI.
- Refuse the IBI and disable interrupts. To do this, the master NACKs to deny the IBI, then sends a Repeated START, and finally sets the DISINT bit in the Command Code Disable Slave Events Command (DISEC) to the interrupting slave.

S	I3C Dynamic Slave Address (RnW=1)	ACK	Slave Byte	T	Sr/P
---	-----------------------------------	-----	------------	---	------

5.26.3.4 I3C private write/read protocol

The primary protocol and mode of I3C is SDR (Single Data Rate) mode. The SDR protocol is based on the I2C standard protocol, with a few notable variations:

- The I3C START and STOP are identical to the I2C START and STOP in their signaling, but they may vary from I2C in their timing.

- The I3C Address Header is identical to the I2C address header in bit form and in signaling, but it may vary from I2C in its timing.
- The data 9-bit words use the same bit count as I2C, but differ in the ninth bit. In I2C, the ninth data bit written by the Master is an ACK by the Slave. By contrast, in I3C the ninth data bit written by the master is the parity of the preceding eight data bits. Therefore, in I3C the slave shall not drive the SDA line for data written by the master in SDR. In I3C read data, this bit allows the Slave to end a read, and allows the master to abort a read.
- In I3C, the SCL line is only driven by the master. Normally this drive is Push-Pull, but it can also be Open Drain.

An I3C message is defined as everything from the initial START (or repeated START) to the next repeated START or STOP. An I3C message is an SDR message if:

- The address in the address header is 7'h7E (the I3C broadcast address). All I3C slaves shall match address value 7'h7E. No I2C slave will match address 7'h7E, because that value is reserved and unused in I2C.
- The address in the address header matches the slave's dynamic address. All I3C slaves shall match their own dynamic address. (It is permitted to then NACKs the header if needed.)

All I3C slaves shall ignore all messages with addresses other than 7'h7E or the I3C master assigned address, and await either the repeated START or the STOP. I3C slaves shall not transmit on the bus in response to a non-matching address.

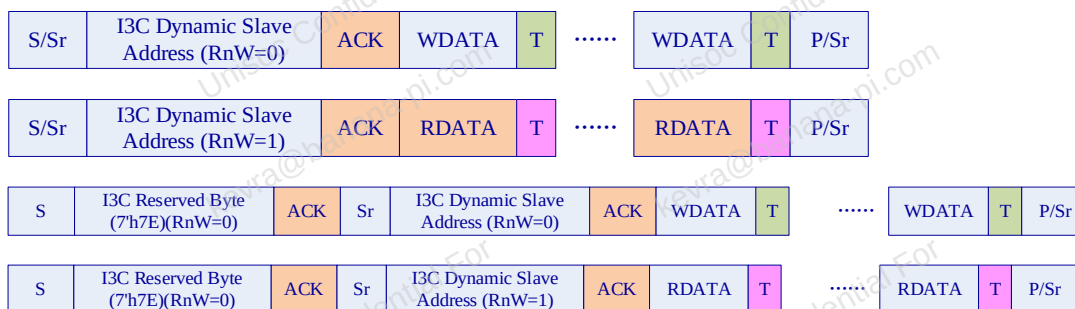


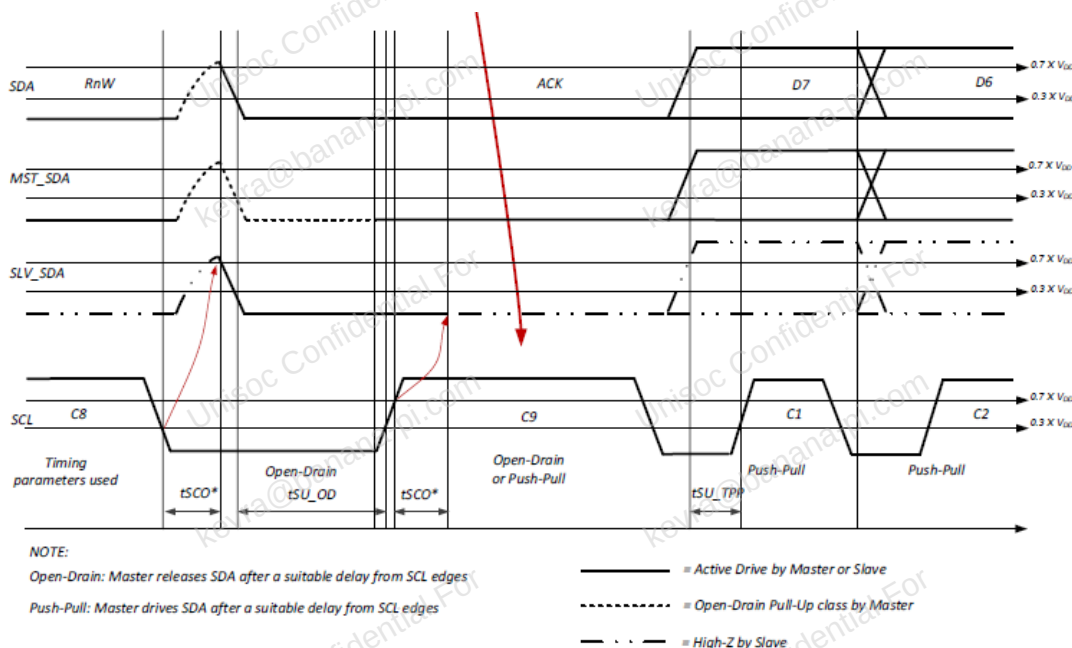
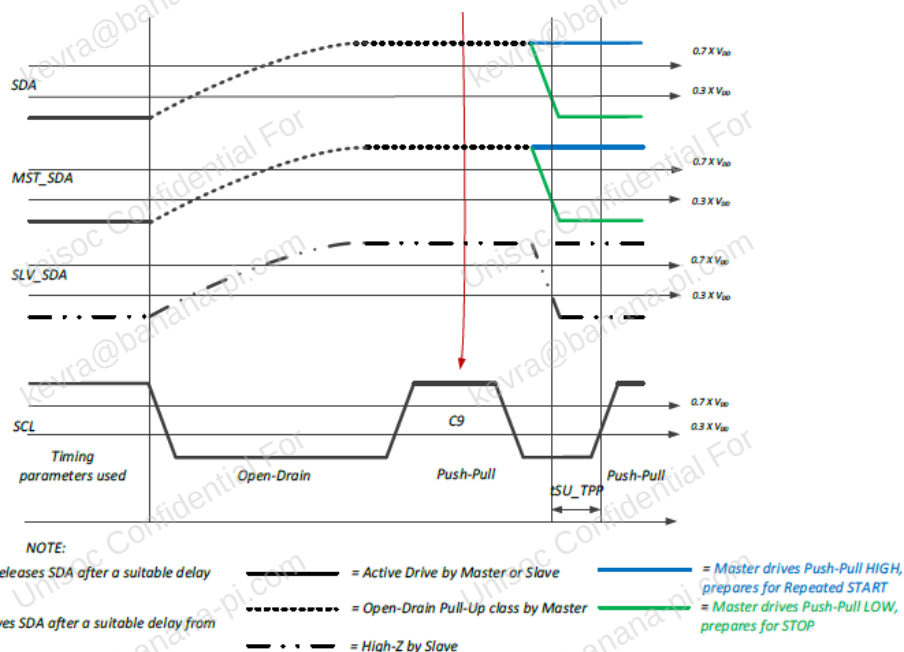
Figure 5-69 I3C Data Transfer – ACK by Slave

Figure 5-70 I3C Data Transfer – NACK by Slave


Figure 5-71 T-Bit When Slave Ends Read and Master Generates STOP

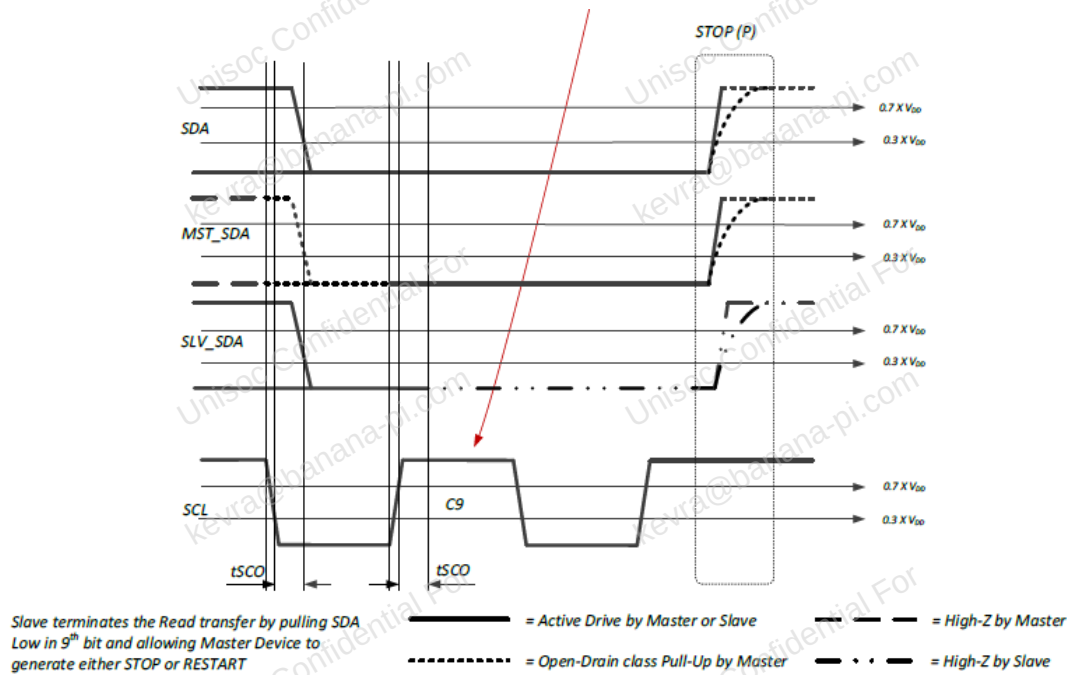


Figure 5-72 T-Bit When Slave Ends Read and Master Generates Repeated START

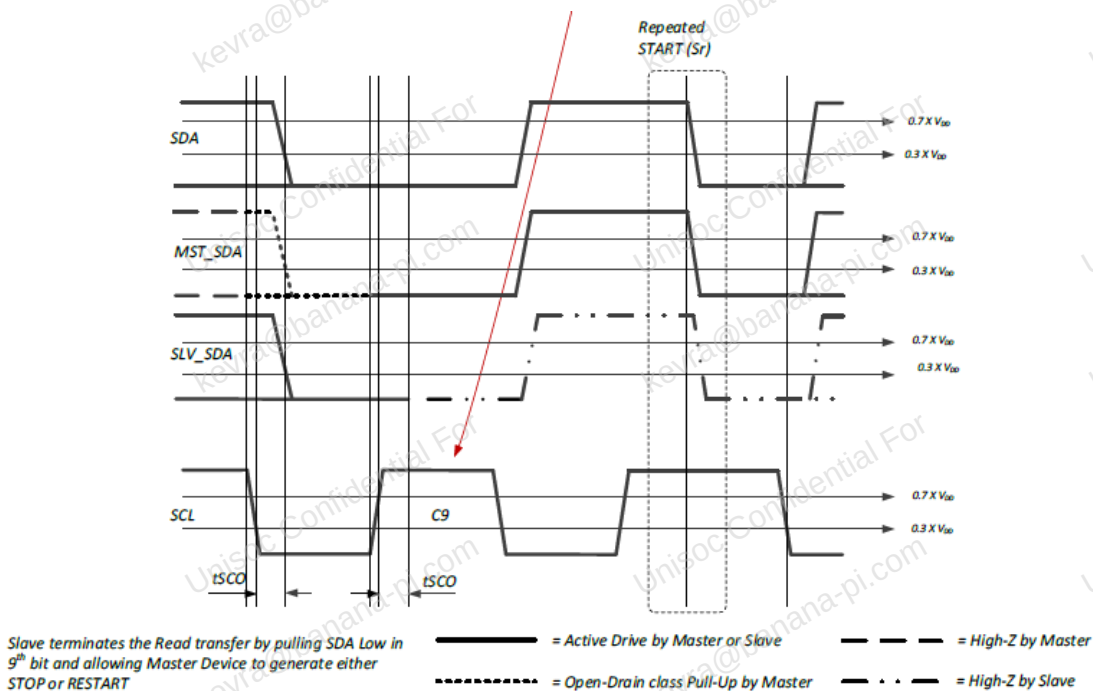


Figure 5-73 T-Bit When Slave and Master Agree to Continue Read Message

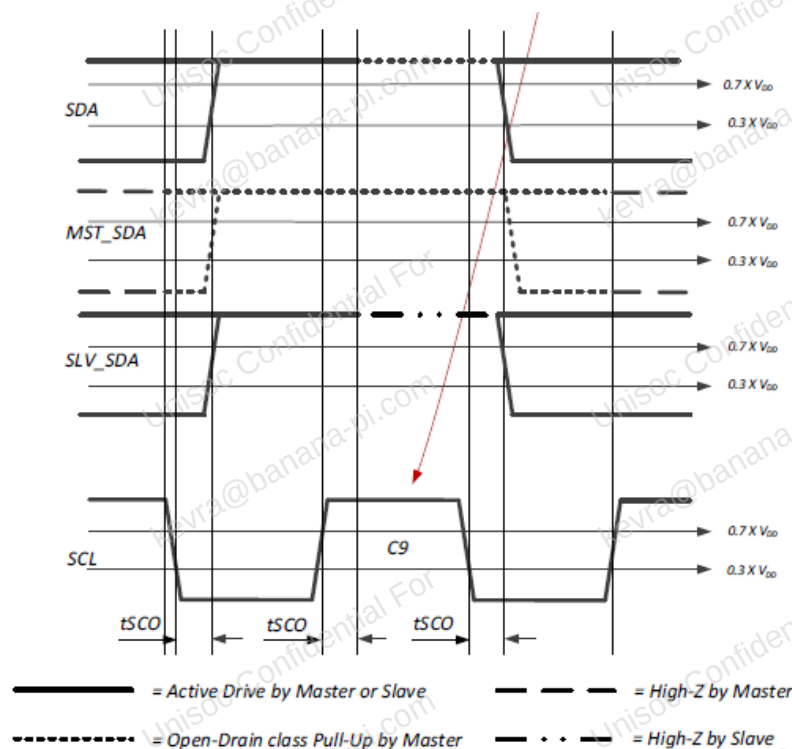


Figure 5-74 T-Bit When Master Ends Read with Repeated START and STOP

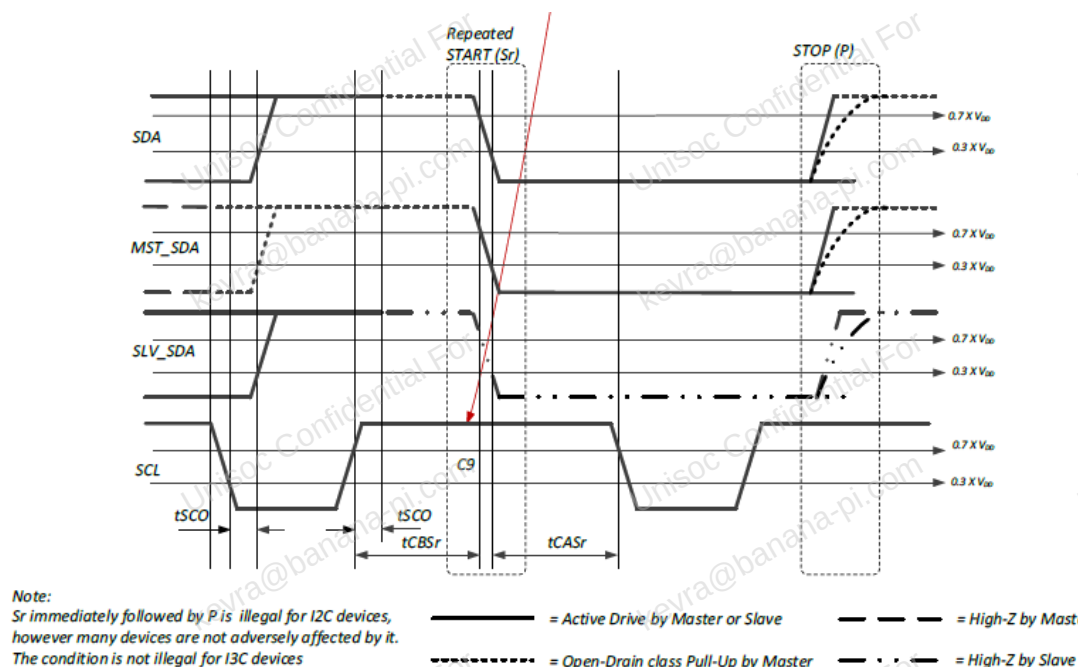
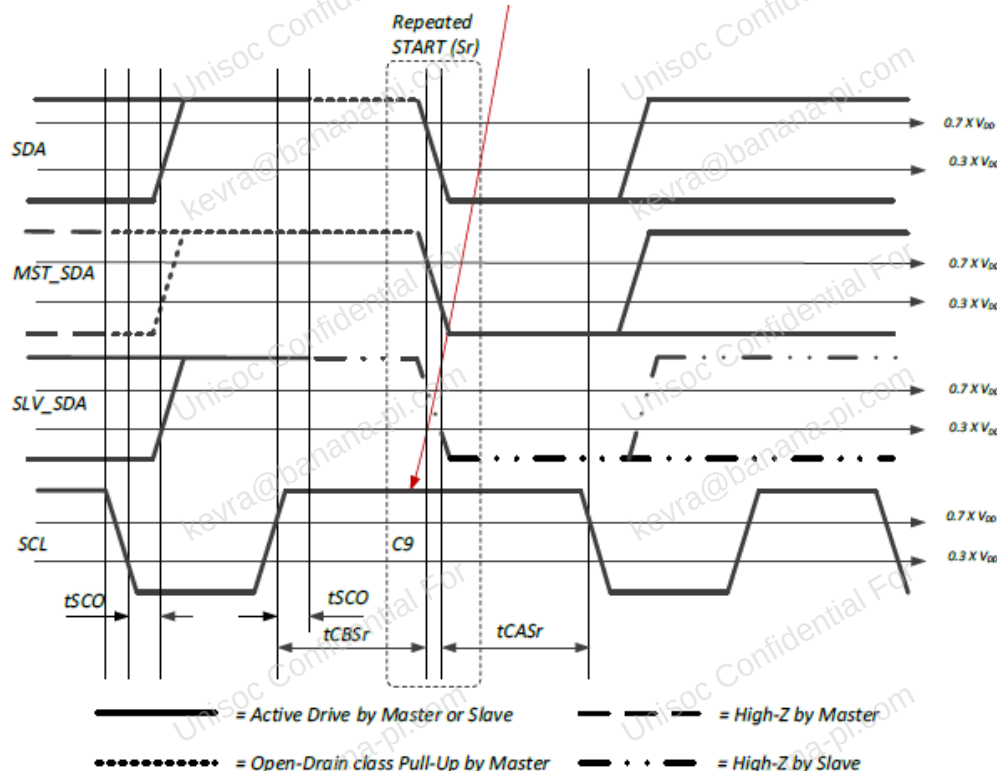


Figure 5-75 T-Bit When Master Ends Read via Repeated START and Further Transfer



5.26.3.5 I3C Address Arbitration

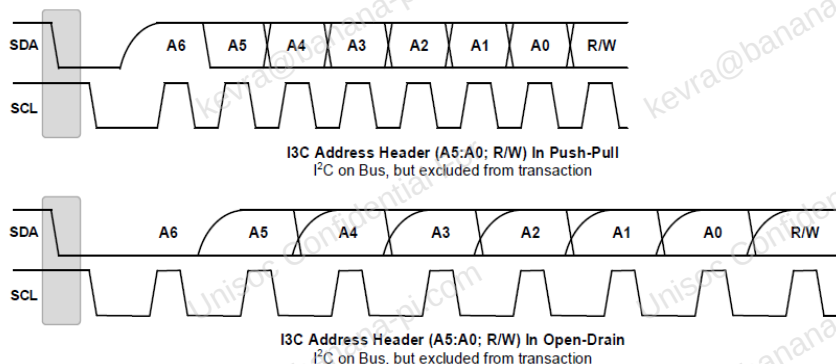
An address header following a START (but not a repeated START) is subject to arbitration, meaning both the master and one or more slaves may attempt to drive an address onto the bus, using SDA. Such address headers are defined as arbitrable address headers. The Arbitration model follows the common Open Drain approach. All devices (whether Master or Slave) that are transmitting an address shall then follow the same rule:

- If the current bit to transmit is a 0, then the device shall drive SDA Low after the falling edge of SCL and hold Low until the next falling edge of SCL.
- If the current bit to transmit is a 1, then the device shall not drive SDA, but rather shall High-Z SDA on the falling edge of SCL.
 - Additionally, the device shall monitor the SDA on the rising edge of SCL to determine whether another device has driven SDA Low.
 - If another device has driven the SDA Low, then the device has “lost” the arbitration and shall not further participate in this address header. That is, the device shall not transmit any more bits, but may wait for a future START Condition (but not a Repeated START Condition).

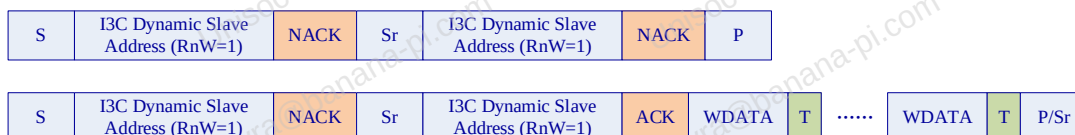
I3C address arbitration may optionally be optimized. As previously described before, an I3C Master Device assigns 7-bit Dynamic Addresses with values in the range $7'h03$ to $7'h7B$. However because the I3C Master treats the entire 9-bit Arbitrable Address Header as Open Drain, it has no way to detect whether a Slave Device might be transmitting its own Address during some(or all) of the Address Header.

Note that for I3C Secondary Masters and I3C In-Band Interrupt Slaves, the I3C Master is free to restrict assigned Dynamic Addresses to the lower half of the available range ($7'h03$ to $7'h3F$), thus leaving the value of Address bit A6 (the first Address bit after the START) as value 0 in all assigned Dynamic Addresses.

If the I3C master is transmitting a 1 value (i.e. High-Z on SDA), as it would when transmitting 7'h7E, then it can monitor SDA on the rising edge of SCL. If SDA has the value 1 (i.e. if SDA is not being driven by any slave), then the I3C master may optionally transmit the remainder of the Address Header (up until the ACK/NACK) in Push-Pull mode.



If the master chooses to start an I3C message with an I3C Dynamic Address, and the addresses match as do the RnW and so neither master nor slave shall ACK since both are expecting the other side to, the master shall transmit the slave address again after a Repeated START (the next one or any one prior to a STOP in the Frame). This allows it to determine which condition occurred and to avoid a deadlock.



5.26.3.6 I2C Mode

I3C is backward compatible with I2C, allowing I2C slave devices to exist on the same devices using the I3C specification. In I2C mode, it supports I2C Messages to Legacy I2C Slaves.

Software flow:

- Set timing parameters configure registers according to I3C clock frequency.
- Set data_cnt.
- Set I3C_TRANS_CTRL configure register.
 - Set i2c_mode = 1'h1.
 - Set i3c_7e_en = 1'h0.
 - Set dma_en = 1'h1 if use DMA mode, otherwise, set dma_en = 1'h0.
 - Set stop_en = 1'h1 if need send STOP condition after transaction finish.
 - Set ccc_en = 1'b0;
 - Set daa_en = 1'b0;
 - Set slv_addr.
 - Set rw_opt.
 - Don't care ccc_cmd.
- Set I3C_TRANS_START configure register.
- Set tx_data then wait trans_end_int if write operation; get rx_data until receive trans_end_int. If data_cnt > 16, software use ae_int and af_int to set tx_data and get rx_data.

5.26.4 Control Registers

See *UMS9620 AON System Control Registers Instructions*.

5.26.5 Application Notes

5.26.5.1 Timing configuration

For I3C, transmission rate is up to 12.5MHz, hence 192M clock is required.

- **Filter_opt is on (default), SCL is 12.0MHz in push-pull, 4.17MHz in open-drain:**
 - Write Timing Para1(0x1C), 32'h1000f
 - Write Timing Para2(0x20), 32'h10001
 - Write Timing Para3(0x24), 32'h6
- **Filter_opt is off, SCL is 12.0MHz in push-pull, 4.17MHz in open-drain:**
 - Write Timing Para1(0x1C), 32'h2000f
 - Write Timing Para2(0x20), 32'h10001
 - Write Timing Para3(0x24), 32'h6

And for I2C, 26M clock meets the requirement.

- **Standard speed (filter_opt is 1, 101.8kHz):**
 - Write Timing Para1(0x1C), 32'h3e003e
 - Write Timing Para2(0x20), 32'h003e003e
 - Write Timing Para3(0x24), 32'h80
- **Standard speed (filter_opt is 0, 98.6kHz):**
 - Write Timing Para1(0x1C), 32'h3f003f
 - Write Timing Para2(0x20), 32'h3f003f
 - Write Timing Para3(0x24), 32'h80
- **Fast speed (filter_opt is 1, 401.7kHz):**
 - Write Timing Para1(0x1C), 32'hd000d
 - Write Timing Para2(0x20), 32'hd000d
 - Write Timing Para3(0x24), 32'h80
- **Fast speed (filter_opt is 0, 401.7kHz):**
 - Write Timing Para1(0x1C), 32'he000e
 - Write Timing Para2(0x20), 32'he000e
 - Write Timing Para3(0x24), 32'h80

5.26.5.2 Hot_join

1. Detecting, and receive i3cm interrupt, read I3C_INT_RAW(0x38), if it is 32'h10(hot-join interrupt), clear interrupt.
2. Write I3C_TRANS_CTRL(0x00), 32'h0 (master ACK hot_join request from slave).
3. Write I3C_TRANS_START(0x04), 32'h1.
4. Receive i3cm interrupt, if I3C_INT_RAW & 32'h1 == 32'h01 (trans_end interrupt), clear interrupt, hot-join request end.

- SW should do DAA(Dynamic Address Assign) in normal case, providing a dynamic address to the hot-joined device.

5.26.5.3 DAA

- Write I3C_INT_EN (0x34), 32'h1f9 (disable ae_int_en, af_int_en, it depends).
- Timing parameter setting, according to the working mode.
- Write I3C_TRANS_CTRL (0x00), 32'h4010, (set daa_en = 1'b1).
- Write I3C_TRANS_START (0x04), 32'h1, start the transaction.
- Receive Slave IDs and store, until a slv_id_int (I3C_INT_RAW is 32'h80), clear the interrupt.
- Write the dynamic address to I3C_TX_DATA(0x18) for each slave.

(may repeat multi times if there are more than one I3C slaves)

Note: Because in I3C slave address assignment has some restrictions so slave may return a NACK if an unexpected address is provided. Then it occurs interrupt called dyc_adr_nack_int (I3C_INT_RAW is 32'h100).

In this case, SW could clear the interrupt and continue to DAA. And store the slave ID which NACKs the provided address.

- Finally, a nack_int is received (I3C_INT_RAW is 32'h20), clear the interrupt.

5.26.5.4 DAA in DMA mode

- Write I3C_INT_EN (0x34), 32'h1f9 (disable ae_int_en, af_int_en, it depends).
- Timing parameter setting, according to the working mode.
- Write I3C_TRANS_CTRL (0x00), 32'h1804010, (set daa_en = 1'b1, daa_dmar_en = 1'b1, daa_dmaw_en = 1'b1).
- Write I3C_DATA_COUNT (0x10), 32'h1.
- Write I3C_TRANS_START (0x04), 32'h1, start the transaction.
- Receive Slave IDs, wait dma_req_i3cm_rx = 1, read Slave IDs from I3C_RX_DATA. If there is a slv_id_int (I3C_INT_RAW is 32'h80), clear the interrupt.
- Wait dma_req_i3cm_tx = 1, write the dynamic address to I3C_TX_DATA(0x18) for each slave. (may repeat multi times if there are more than one I3C slaves)
- Finally, a nack_int is received (I3C_INT_RAW is 32'h20), clear the interrupt.

5.26.5.5 IBI

IBI occurs after DAA is complemented, and each slave has its own dynamic address.

- Receive i3cm interrupt, if it is IBI interrupt, (I3C_INT_RAW is 32'h8), clear interrupt.
- Write I3C_TRANS_CTRL (0x00), 32'h10. (Set MST_ACK = 0, will ACK IBI).
- If slave's BCR [2] bit is set to 1, write 16'hFFFF to I3C_DATA_COUNT (0x10), it enables the slave to send enough bytes and abort the transaction actively. Else write 16'hFFFF to I3C_DATA_COUNT(0x10), which means no data follows the IBI.
- Write I3C_TRANS_START (0x04), 32'h1, start the transaction.
- Receive a trans_end interrupt (I3C_INT_RAW is 32'h1), clear the interrupt.

5.26.5.6 I3C Private Write

I3C Write/Read should do after DAA is finished.

1. Write I3C_INT_EN (0x34), 32'h1f9 (should enable ae_int if data_cnt > 16).
2. Timing parameter setting, according to the working mode.
3. Write I3C_TRANS_CTRL (0x0), (Set i3c_7e_en = 1'b1, stop_en = 1'b1, rw_opt = 1'b0, slv_addr is the address assigned in DAA)
4. Write I3C_DATA_COUNT (0x10) as required.
5. Write I3C_TX_DATA (0x18) as required.
6. Write I3C_TRANS_START (0x0), 32'h1.
7. Wait interrupt (I3C_INT_RAW = 32'h01), clear interrupt.

5.26.5.7 I3C Private Read

I3C Write/Read should do after DAA is finished.

1. Write I3C_INT_EN (0x34), 32'h1f9 (should enable af_int if data_cnt > 16).
2. Timing parameter setting, according to the working mode.
3. Write I3C_TRANS_CTRL (0x0), (Set i3c_7e_en = 1'b1, stop_en = 1'b1, rw_opt = 1'b1, slv_addr is the address assigned in DAA)
4. Write I3C_DATA_COUNT (0x10) as required.
5. Write I3C_TRANS_START (0x0), 32'h1.
6. Wait interrupt (I3C_INT_RAW = 32'h01), clear interrupt.
7. Read I3C_RX_DATA (0x14) as required.

5.26.5.8 I3C Private Write in DMA mode

I3C Write/Read should do after DAA is finished.

1. Write I3C_INT_EN (0x34), 32'h1f9 (should enable ae_int if data_cnt > 16).
2. Timing parameter setting, according to the working mode.
3. Write I3C_TRANS_CTRL (0x0), (Set i3c_7e_en = 1'b1, stop_en = 1'b1, dma_en = 1'b1, rw_opt = 1'b0, slv_addr is the address assigned in DAA)
4. Write I3C_DATA_COUNT (0x10) as required.
5. Write I3C_TRANS_START (0x0), 32'h1.
6. Wait dma_req_i3cm_tx, write I3C_TX_DATA (0x18) as required.
7. Wait interrupt (I3C_INT_RAW = 32'h01), clear interrupt.

Note: DMA mode supports i2c write.

5.26.5.9 I3C Private Read in DMA mode

I3C Write/Read should do after DAA is finished.

1. Write I3C_INT_EN (0x34), 32'h1f9 (should enable af_int if data_cnt > 16).
2. Timing parameter setting, according to the working mode.
3. Write I3C_TRANS_CTRL (0x0), (Set i3c_7e_en = 1'b1, stop_en = 1'b1, dma_en = 1'b1, rw_opt = 1'b1, slv_addr is the address assigned in DAA)
4. Write I3C_DATA_COUNT (0x10) as required.
5. Write I3C_TRANS_START (0x0), 32'h1.
6. Wait interrupt (I3C_INT_RAW = 32'h01), clear interrupt.
7. Read I3C_RX_DATA (0x14) as required.

Note: DMA mode supports i2c read.

5.26.5.10 I2C Write

1. Write I3C_INT_EN (0x34), 32'h1f9 (should enable ae_int if data_cnt > 16).
2. Timing parameter setting, according to the working mode.
3. Write I3C_TRANS_CTRL (0x0), (Set i3c_7e_en = 1'b0, stop_en = 1'b1, rw_opt = 1'b0, i2c_en = 1'b1, slv_addr is the address of corresponding slave)
4. Write I3C_DATA_COUNT (0x10) as required.
5. Write I3C_TX_DATA (0x18) as required.
6. Write I3C_TRANS_START (0x0), 32'h1.
7. Wait trans_end interrupt (I3C_INT_RAW = 32'h01), clear interrupt.

5.26.5.11 I2C Read

1. Write I3C_INT_EN (0x34), 32'h1f9 (should enable ae_int if data_cnt > 16).
2. Timing parameter setting, according to the working mode.
3. Write I3C_TRANS_CTRL (0x0), (Set i3c_7e_en = 1'b0, stop_en = 1'b1, rw_opt = 1'b1, i2c_en = 1'b1, slv_addr is the address of corresponding slave)
4. Write I3C_DATA_COUNT (0x10) as required.
5. Write I3C_TRANS_START (0x0), 32'h1.
6. Wait trans_end interrupt (I3C_INT_RAW = 32'h01), clear interrupt.
7. Read I3C_RX_DATA (0x18) as required.

5.26.5.12 BROAD_CCC_WRITE

1. Write I3C_INT_EN (0x34), 32'h1f9 (disable ae_int_en, af_int_en, it depends).
2. Timing parameter setting, according to the working mode.
3. Write I3C_TRANS_CTRL (0x0), (Set i3c_7e_en = 1'b1, stop_en = 1'b1, ccc_en = 1'b1, rw_opt = 1'b0, ccc_cmd = 8'h02 .e.g.)
4. Write I3C_TX_DATA (0x18) as required, 1byte only.
5. Write I3C_TRANS_START (0x0), 32'h1.
6. Wait trans_end interrupt (I3C_INT_RAW = 32'h01), clear interrupt.

5.26.5.13 DIRECT_CCC_WRITE

1. Write I3C_INT_EN (0x34), 32'h1f9 (disable ae_int_en, af_int_en, it depends).
2. Timing parameter setting, according to the working mode.
3. Write I3C_TRANS_CTRL (0x0), (Set i3c_7e_en = 1'b1, stop_en = 1'b1, ccc_en = 1'b1, rw_opt = 1'b0, slv_addr is the assigned address in DAA, ccc_cmd = 8'h80 .e.g.)
4. Write I3C_TX_DATA (0x18) if required, 1byte only.
5. Write I3C_TRANS_START (0x0), 32'h1.
6. Wait trans_end interrupt (I3C_INT_RAW = 32'h01), clear interrupt.

5.26.5.14 DIRECT_CCC_READ

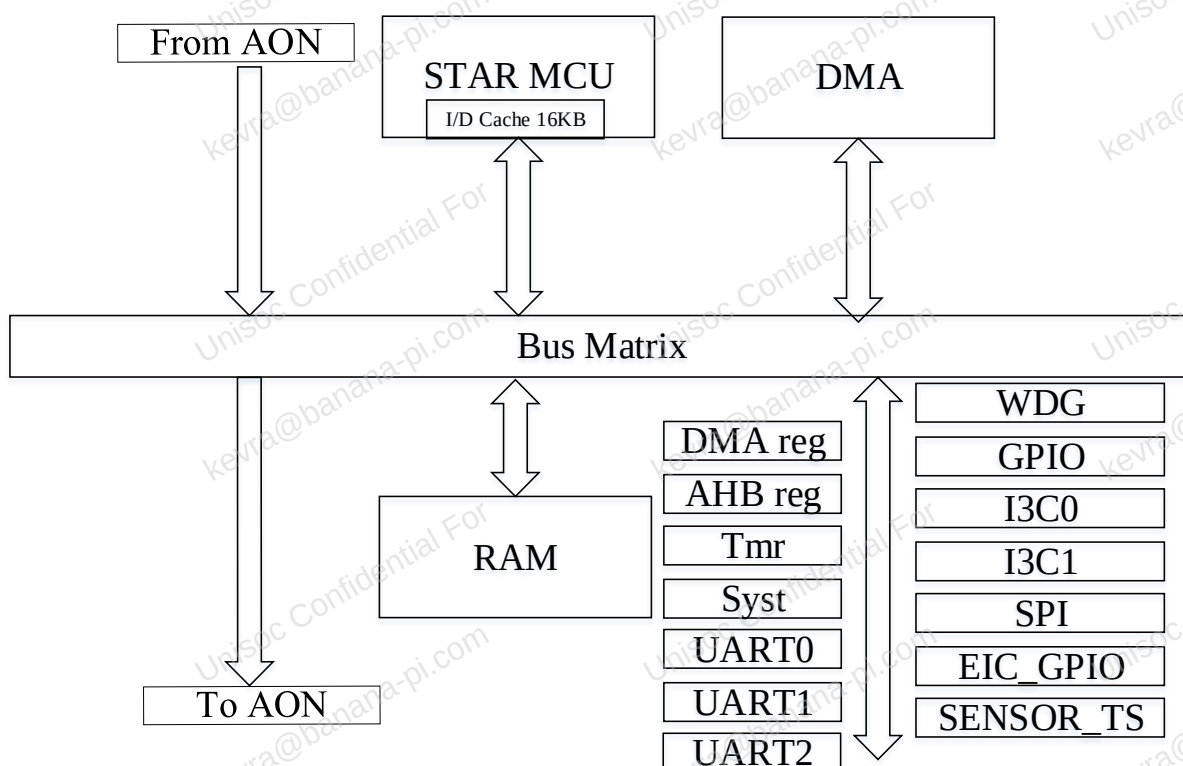
1. Write I3C_INT_EN (0x34), 32'h1f9 (disable ae_int_en, af_int_en, it depends).

2. Timing parameter setting, according to the working mode.
3. Write I3C_TRANS_CTRL (0x0), (Set i3c_7e_en = 1'b1, stop_en = 1'b1, ccc_en = 1'b1, rw_opt = 1'b1, slv_addr is the assigned address in DAA, ccc_cmd = 8'h8D .e.g.)
4. Write I3C_DATA_COUNT (0x10) as required.
5. Write I3C_TRANS_START (0x0), 32'h1.
6. Wait trans_end interrupt (I3C_INT_RAW = 32'h01), clear interrupt.
7. Read I3C_RX_DATA (0x14), it depends.

6 CH Subsystem

6.1 Functional Block Diagram

Figure 6-1 The Functional Block Diagram of CH(Context Hub)



6.2 Features

- Arm® China STAR MCU-SE core
 - Cortex®-M v8 architecture
 - Arm®v8-M DSP extension supported
 - Single-precision floating-point
 - 16 memory regions
 - Up to 192MHz
 - Instruction cache size is 16KB
 - Data cache size is 16KB

- Arm® AMBA® 4 AHB bus interface
- 384KB On chip SRAM
- DMA Controller support fast data transfers to off-load the processor
- 64-bit/32-bit decreasing counter System Timer
- Watchdog support 32-bit timer value loading
- EIC module includes four sub-modules: EIC_DBNC, EIC_LATCH, EIC_ASYNC and EIC_SYNC
- I3C interface is intended to improve upon the feature of the I2C interface, preserving backward compatibility
- Full-duplex UART
- SPI support MICRO/PLUS mode

6.3 System Addr mapping

Table 6-1 The ch Address map

Module	Type	ch_star address	apcpu address
CH_RAM	Mem	0x0000_0000~0x0005_FFFF	0x6501_6000~0x6507_5FFF
CH_TMR	Reg	0x6220_0000~0x6220_FFFF	0x6220_0000~0x6220_FFFF
CH_SYST	Reg	0x6221_0000~0x6221_FFFF	0x6221_0000~0x6221_FFFF
CH_WDG	Reg	0x6222_0000~0x6222_FFFF	0x6222_0000~0x6222_FFFF
CH_UART0	Reg	0x6223_0000~0x6223_FFFF	0x6223_0000~0x6223_FFFF
CH_UART1	Reg	0x6224_0000~0x6224_FFFF	0x6224_0000~0x6224_FFFF
CH_UART2	Reg	0x6225_0000~0x6225_FFFF	0x6225_0000~0x6225_FFFF
CH_GPIO	Reg	0x6227_0000~0x6227_FFFF	0x6227_0000~0x6227_FFFF
CH_I3C0	Reg	0x6228_0000~0x6228_FFFF	0x6228_0000~0x6228_FFFF
CH_I3C1	Reg	0x6229_0000~0x6229_FFFF	0x6229_0000~0x6229_FFFF
CH_SPI	Reg	0x622A_0000~0x622A_FFFF	0x622A_0000~0x622A_FFFF
CH_EIC_GPIO	Reg	0x622B_0000~0x622B_FFFF	0x622B_0000~0x622B_FFFF
CH_DMA	Reg	0x6210_0000~0x6210_FFFF	0x6210_0000~0x6210_FFFF
CH_AHB_REG	Reg	0x6211_0000~0x6211_FFFF	0x6211_0000~0x6211_FFFF

6.4 System Interrupt

Table 6-2 ch_sys interrupt

Signal	Num	Description	Source
int_req_ch_core_soft	44	CH sys core soft interrupt	aon_apb_rf
int_req_ch_STAR MCU_fpu	43	STAR MCU fpu interrupt	ch_cm_star
int_req_ch_sensor_ts3	42	CH sensor interrupt	ch_sensor_ts
int_req_ch_sensor_ts2	41	CH sensor interrupt	ch_sensor_ts
int_req_ch_sensor_ts1	40	CH sensor interrupt	ch_sensor_ts
int_req_ch_sensor_ts0	39	CH sensor interrupt	ch_sensor_ts
outbox_scr_irq_sig[1]	38	out mailbox interrupt	mm_mailbox
inbox_scr_irq_sig[1]	37	in mailbox interrupt	mm_mailbox
int_req_vbc_audply	36	aud vbc play interrupt	audcp_vbc
int_req_vbc_audrcd	35	aud vbc sound recording	audcp_vbc
int_req_mcdt_ap	34	aud mcdt to AP interrupt	audcp_vbc
int_req_dmaap	33	aud dma to AP CPU interrupt	audcp_dma
int_req_aud_vbc	32	aud vbc interrupt	audcp_vbc
int_req_dpm_rx	31	aud pdm voice in interrupt	audcp_aon_pdm
int_req_ch_tmr_tmr0	30	CH sys tmr interrupt	ch_tmr
int_req_ch_tmr_tmr1	29	CH sys tmr interrupt	ch_tmr
int_req_ch_tmr_tmr2	28	CH sys tmr interrupt	ch_tmr
int_req_ch_syst	27	CH sys syst interrupt	ch_syst
int_req_ch_wdg	26	CH sys watchdog interrupt	ch_wdg
int_req_ch_uart0	25	CH sys uart interrupt	ch_uart0
int_req_ch_uart1	24	CH sys uart interrupt	ch_uart1
int_req_ch_uart2	23	CH sys uart interrupt	ch_uart2
int_req_ch_gpio	22	CH sys gpio interrupt	ch_gpio
int_req_ch_i3cm0	21	CH sys i3c interrupt	ch_i3c0
int_req_ch_i3cm1	20	CH sys i3c interrupt	ch_i3c1
int_req_ch_spi	19	CH sys spi interrupt	ch_spi
int_req_ch_eic_gpio_dbnc	18	CH sys eic interrupt	ch_eic_gpio
int_req_ch_eic_gpio_lat	17	CH sys eic interrupt	ch_eic_gpio
int_req_ch_eic_gpio_async	16	CH sys eic interrupt	ch_eic_gpio
int_req_ch_eic_gpio_sync	15	CH sys eic interrupt	ch_eic_gpio

Signal	Num	Description	Source
int_req_ch_dma	14	CH sys dma interrupt	ch_dma
int_req_mailbox_in2	13	in mailbox interrupt	aon_mailbox
int_req_mailbox_in6	12	in mailbox interrupt	aon_mailbox
int_req_mailbox_out2	11	out mailbox interrupt	aon_mailbox
thm0_int	10	CH thermal interrupt	aon_thm0
thm1_int	9	CH thermal interrupt	aon_thm1
thm2_int	8	CH thermal interrupt	aon_thm2
thm3_int	7	CH thermal interrupt	aon_thm3
int_req_sec_cnt_top	6	security cnt interrupt	aon_sec_cnt_top
int_req_sec_tmr	5	security tmr interrupt	aon_sec_tmr
int_req_sec_gpio	4	security gpio interrupt	aon_sec_gpio
int_req_sec_eic	3	security eic interrupt	aon_sec_eic
int_clk_32k_det	2	clk_32k_det interrupt	aon_clk_32k_det
int_req_irq_n	1	apcpu interrupt to ch	aon_lp_ap_intc
int_req_fiq_n	0	apcpu interrupt to ch	aon_lp_ap_intc

6.5 System Low power Design

CH STAR MCU Power Design

6.5.1 Power Scheme

CH sys is on the always on power domain, so CH sys will never power off. But there are 3 power states in CH STAR MCU: active, light sleep, deep sleep.

Table 6-3 CH STAR MCU power states information

Power Status	Modules
Active	STAR MCU Core Active
light sleep	Only STAR MCU Core sleep
deep sleep	CH sys sleep

6.5.2 Boot Information

APCPU has to set CH system clock/reset reg for boot.

STAR MCU Boot related register			
AON PMU APB REG (Base Addr 0x6491_0000)			
Name	Offset	Wr Value	Description
FORCE_DEEP_SLEEP_CFG_0	0x0818	Bit[9]=0	bit[9]= CH_FORCE_DEEP_SLEEP_REG default=1,set/clear Force CH deep sleep; 0: not force; 1: force

APCPU copy STAR MCU code from DDR to STAR MCU RAM, then release STAR MCU core soft reset related register.

NOTE

CH_CORE_SOFT_RST default value is 1 and high active,APCPU has to make sure the CH sys is in reset state during code movement, and then release CH_CORE_SOFT_RST to 0.

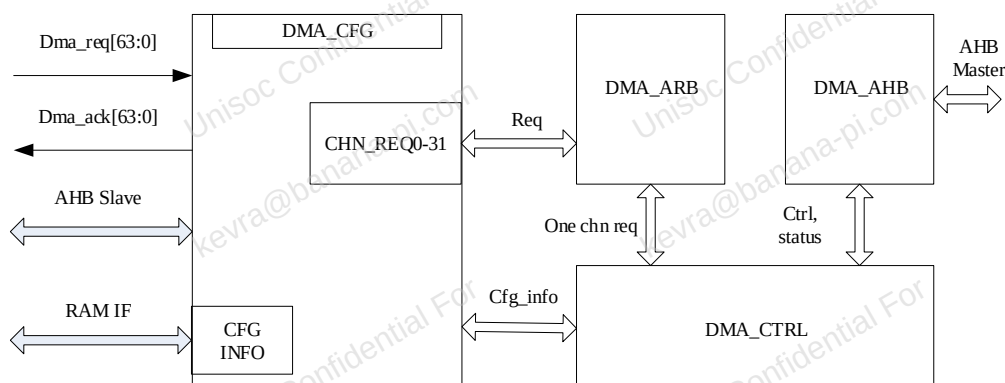
STAR MCU Boot related register			
AON APB REG (Base Addr 0x6490_0000)			
Name	Offset	Wr Value	Description
CH_SYS_SOFT_RST	0x8C	Bit[0]=0	bit[0]= CH_CORE_SOFT_RST,default=1,set/clear
		Bit[4]=0	bit[4]= CH_SYS_SOFT_RST,default=0,set/clear

6.6 DMA controller

6.6.1 Overview

AON has one AHB interface DMA which is used to offload MCU loading, and reduce power consumption.

Figure 6-2 AHB DMA Arch



6.6.2 Features

- AMBA® 2 AHB Specification.
- Two 32-bit AHB masters for read and write operation.
- One AHB 32-bit slave for configuration only supports word access to AHB slave port.
- Support Byte/Half-word/word transfer on AHB Read/Write Master.
- Not support Arm® AMBA® ERROR/RETRY/SPLIT response.
- Support Arm® AMBA® NSEQ/INCR4/INCR8 transaction, INCR4/INCR8 configurable.
- AHB Master Big-endian or Little-endian.
- Support 32 channels assigned to hardware request by Channel ID;
- One physical channel shared by 32 Logic channels.
- 4-level configurable priority. 3 has the highest and 0 has the lowest priority. For the channels with the same priority, round-robin would be preferred.
- A-synchronized/Synchronized hand-shake request & acknowledge.
- Accept request from both hardware and software.
- Un-aligned word accesses, for both read and write operation.
- Byte/Half-word switch scheme.
- Support link list for scatter and gather transfer.
- Configurable fragment wait time between two successive fragments.
- One request for fragment/block/transaction/link-list.
- Automatically disable channel after requests done.
- Support 128K-1 Byte fragment length.
- Support 128K-1 Byte block length (Block length may be less than fragment length in Standard Channel, while forbidden in Full channel).
- Support 256M-1 Byte transaction length.
- Support transfer step, ranged from – (32K-1) Bytes to (32K-1) Bytes, only for full DMA channel.
- Support fragment step, ranged from – (32K-1) Bytes to (32K-1) Bytes, only for full DMA channel.
- Support block step, ranged from – (128M-1) Bytes to (128M-1) Bytes, only for full DMA channel.
- Address-fix transfer for either Source side or destination side.
- Address-wrap transfer for either Source side or destination side.
- Pause DMA or pause logical channel.
- Interrupt of fragment, block, transaction and link list, with raw and masked status.
- Interrupt of Error configuration, and support errors status.
- Support hardware request to be re-allocated by Channel ID.
- Support automatically sleep when all requests served, and wakeup when hardware request active.
- Support software retention if power domain of DMA shut down.

6.6.3 Function Descriptions

General Description

DMA Controller support fast data transfers to off-load the processor. With DMA, specific devices can complete quick data movement from/to memories, internal RAM or External SRAM. DMA Controller can also be used to connect any two devices other than memories, as long as they can be addressed in memory space.

6.6.4 Control Registers

See *UIS7885 AON System Control Registers Instructions*.

6.6.5 Application Notes

Table 6-4 AON DMA Request List

hardware/software channel request	DMA Request List	Request Source	Type	NUM
Hardware Channel Request		63~21 unused		
	dma_req_aud_pdm_rx_ch	sys_cp_aud	Full	20
	dma_req_aud_vbc_adc7_ch	sys_cp_aud	Full	19
	dma_req_aud_vbc_adc6_ch	sys_cp_aud	Full	18
	dma_req_aud_vbc_adc5_ch	sys_cp_aud	Full	17
	dma_req_aud_vbc_adc4_ch	sys_cp_aud	Full	16
	dma_req_aud_vbc_adc3_ch	sys_cp_aud	Full	15
	dma_req_aud_vbc_adc2_ch	sys_cp_aud	Full	14
	dma_req_aud_vbc_adc1_ch	sys_cp_aud	Full	13
	dma_req_aud_vbc_adc0_ch	sys_cp_aud	Full	12
	dma_req_ch_uart2_rx	ch_uart2	Full	11
	dma_req_ch_uart2_tx	ch_uart2	Full	10
	dma_req_ch_uart1_rx	ch_uart1	Full	9
	dma_req_ch_uart1_tx	ch_uart1	Full	8
	dma_req_ch_uart0_rx	ch_uart0	Full	7
	dma_req_ch_uart0_tx	ch_uart0	Full	6
	dma_req_ch_i3c1_rx	ch_i3c1	Full	5
	dma_req_ch_i3c1_tx	ch_i3c1	Full	4
	dma_req_ch_i3c0_rx	ch_i3c0	Full	3
	dma_req_ch_i3c0_tx	ch_i3c0	Full	2
	dma_req_ch_spi_rx	ch_spi	Full	1
	dma_req_ch_spi_tx	ch_spi	Full	0
Software Channel Request	backup	1		

6.7 Sensor Time Stamp

6.7.1 Overview

Synchronization of time counters between CH STAR and APCPU.

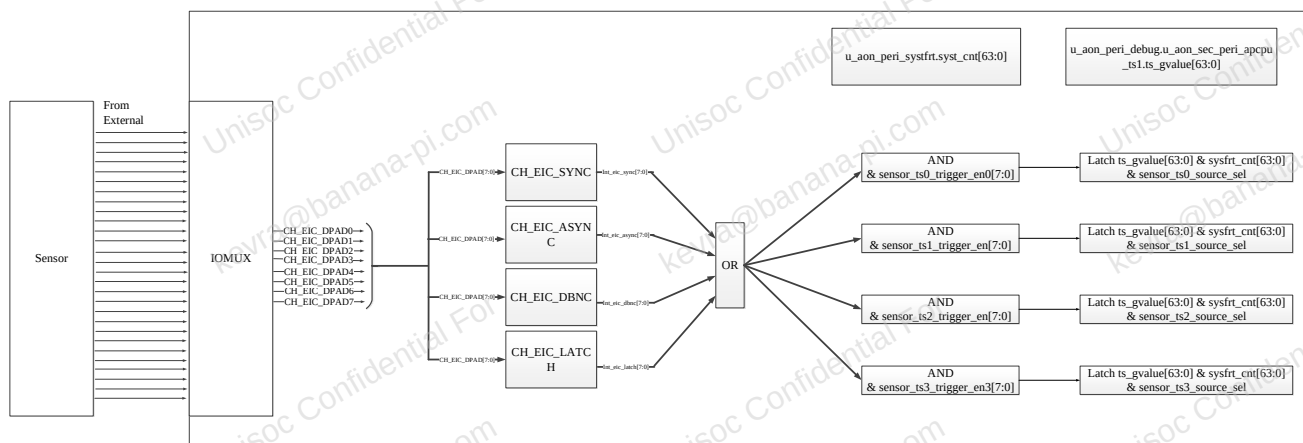
6.7.2 Function Descriptions

IOMUX module select 8 paths to EIC_CH_DPAD[7:0] from external PAD. The 8 paths are completely independent. Software can set trigger condition for each path.

The software can set each path in CH_AHB_RF. After CH_EIC processing, 8 outputs are connected to CH_AHB_RF four groups of registers, software can set trigger conditions, namely trigger_en[7:0].

When trigger_en satisfies the trigger condition, CH_AHB_RF will latch aon_peri_sysftr. syst_cnt[63:0], aon_peri_debug. u_aon_sec_peri_apcpu_ts1. ts_gvalue[63:0] and related source_sel. A total of 129bits of data. That is, the 129bits read-only registers in each group record the latch value at the last trigger time.

Figure 6-3 SENSOR TS Arch



6.7.3 Application Notes

First of all, the software needs to enable sysftr and APCU_TS1, follow up process:

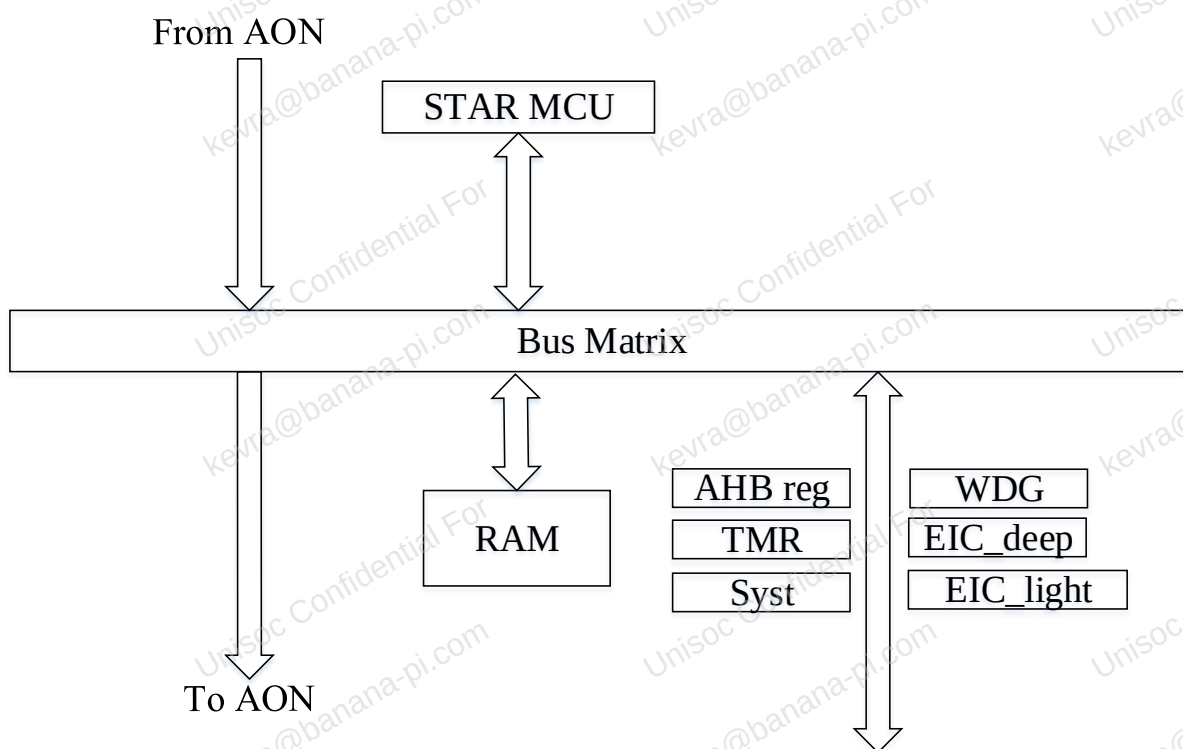
1. Enable **ch_eic_gpio_top**, then set trigger condition.
2. Enable **sensor_ts**, and set **source_sel** and **trigger_en** in **CH_AHB_RF**.
3. Wait trigger event, when trigger happens, then enter **int_sensor_ts** interrupt service of **ch STAR MCU**.
4. Clear **eic** interrupt, and then exit the interrupt service.
5. Software can get the 129bits latch value in **CH_AHB_RF**.

7

SP Subsystem

7.1 Functional Block Diagram

Figure 7-1 The Functional Block Diagram of SP(System Processor)



7.2 Features

- Arm® China STAR MCU-CU core
 - Cortex®-M v8 architecture
 - 16 memory regions
 - Up to 128MHz
- Arm® AMBA® 4 AHB bus interface
- 64KB On chip SRAM
- 64-bit/32-bit decreasing counter System Timer
- Watchdog support 32-bit timer value loading
- EIC module includes four sub-modules: EIC_DBNC, EIC_LATCH, EIC_ASYNC and EIC_SYNC

7.3 System Addr mapping

Table 7-1 The sp Address map

Module	Type	sp_star address	apcpu/ch_star address
SP_RAM	Mem	0x0000_0000~0x0000_FFFF	0x6280_0000~0x6280_FFFF
SP_AHB_REG	Reg	0x6291_0000~0x6291_FFFF	0x6291_0000~0x6291_FFFF
SP_TMR	Reg	0x62A0_0000~0x62A0_FFFF	0x62A0_0000~0x62A0_FFFF
SP_SYST	Reg	0x62A1_0000~0x62A1_FFFF	0x62A1_0000~0x62A1_FFFF
SP_WDG	Reg	0x62A2_0000~0x62A2_FFFF	0x62A2_0000~0x62A2_FFFF
SP_EIC_DEEP	Reg	0x62AB_0000~0x62AB_FFFF	0x62AB_0000~0x62AB_FFFF
SP_EIC_LIGHT	Reg	0x62AC_0000~0x62AC_FFFF	0x62AC_0000~0x62AC_FFFF

7.4 System Interrupt

Table 7-2 sp_sys interrupt

Signal	Num	Description	Source
int_req_ese_wdg_rst_flag	71	ise watchdog interrupt	sys_ise
int_req_phycp_wdg_rst_flag	70	phycp watchdog interrupt	sys_cp_phy
int_req_busmon_tmc_axi	69	dbg sys Bus Monitor interrupt	sys_dbg
int_req_pcie_pwr_up	68	pcie sys power up interrupt	pmu
int_req_ipa_pwr_up	67	ipa sys power up interrupt	pmu
int_req_busmon_ddc_axi	66	dbg sys ddc Bus Monitor interrupt	sys_dbg
etr_axi_mon_int	65	dbg sys monitor interrupt	sys_dbg
thm3_int	64	thermal sensor interrupt	sys_aon
thm2_int	63	thermal sensor interrupt	sys_aon
thm1_int	62	thermal sensor interrupt	sys_aon
thm0_int	61	thermal sensor interrupt	sys_aon
int_req_aud_busmon	60	aud Bus Monitor interrupt	sys_aud_cp
int_req_acc_prot_pmu	59	acc protect pmu interrupt	sys_aon
int_req_acc_prot_aon_apb_reg	58	acc protect aon_apb_rf interrupt	sys_aon
int_pub_dmc_mpu_vio	57	pub sys dmc_mpu_vio interrupt	sys_pub
int_pub_ptm	56	pub sys ptm monitor interrupt	sys_pub

Signal	Num	Description	Source
int_pub_dfi_busmon	55	pub sys dfi Bus Monitor interrupt	sys_pub
int_pub_dfs_giveup	54	pub sys dfs giveup interrupt	sys_pub
int_pub_dfs_error	53	pub sys dfs error interrupt	sys_pub
int_pub_dfs_vote_done	52	pub sys dfs vote done interrupt	sys_pub
int_pub_dfs_deny	51	pub sys dfs deny interrupt	sys_pub
int_pub_dfs_exit	50	pub sys dfs exit interrupt	sys_pub
int_pub_dfs_complete	49	pub sys dfs complete interrupt	sys_pub
int_req_sp_core_soft	48	sp core soft interrupt	aon_apb_rf
int_req_busmon_ufshc	47	AP sys Bus Monitor ufshc	sys_ap
int_req_busmon_dbg_iram	46	ipa sys Bus Monitor interrupt	sys_ipa
int_req_busmon_ipa_m0	45	ipa sys Bus Monitor interrupt	sys_ipa
int_req_busmon_pam_wifi	44	ipa sys Bus Monitor interrupt	sys_ipa
int_req_busmon_pam_u3	43	ipa sys Bus Monitor interrupt	sys_ipa
busmon_int_dcaml_blk	42	camera sys dcaml interrupt	sys_ap_mm
int_req_aud_wdg_rst_extd	41	aud sys watchdog interrupt	sys_cp_aud
int_req_vbc_audply	40	aud vbc play interrupt	sys_cp_aud
int_req_vbc_audrcd	39	aud vbc sound recording	sys_cp_aud
int_req_mcdt_ap	38	aud mcdt to AP interrupt	sys_cp_aud
int_req_dmaap	37	aud dma to apcpu interrupt	sys_cp_aud
dma_chn2_grp2_int	36	aud dma interrupt	sys_cp_aud
dma_chn1_grp2_int	35	aud dma interrupt	sys_cp_aud
dma_chn2_grp1_int	34	aud dma interrupt	sys_cp_aud
dma_chn1_grp1_int	33	aud dma interrupt	sys_cp_aud
int_req_phycp_busmon	32	phycp sys Bus Monitor interrupt	sys_cp_phy
pscp_busmon_irq	31	pscp sys Bus Monitor interrupt	sys_cp_ps
int_req_ch_wdg_rst_extd	30	CH sys watchdog interrupt	ch_sys
int_req_sp_tmr_tmr0	29	SP sys tmr interrupt	sp_tmr_top
int_req_sp_tmr_tmr1	28	SP sys tmr interrupt	sp_tmr_top
int_req_sp_tmr_tmr2	27	SP sys tmr interrupt	sp_tmr_top
int_req_sp_syst	26	SP sys syst interrupt	sp_syst
int_req_sp_wdg	25	SP sys watchdog interrupt	sp_wdg

Signal	Num	Description	Source
int_req_sp_eic_light_dbnc	24	SP sys eic interrupt	sp_eic_light
int_req_sp_eic_light_lat	23	SP sys eic interrupt	sp_eic_light
int_req_sp_eic_light_async	22	SP sys eic interrupt	sp_eic_light
int_req_sp_eic_light_sync	21	SP sys eic interrupt	sp_eic_light
int_req_sp_eic_deep_dbnc	20	SP sys eic interrupt	sp_eic_deep
int_req_sp_eic_deep_lat	19	SP sys eic interrupt	sp_eic_deep
int_req_sp_eic_deep_async	18	SP sys eic interrupt	sp_eic_deep
int_req_sp_eic_deep_sync	17	SP sys eic interrupt	sp_eic_deep
int_req_pscp_wdg_rst_flag	16	watchdog in aon sys interrupt	sys_aon
int_req_mailbox_in1	15	in mailbox interrupt	sys_aon
int_req_mailbox_out1	14	out mailbox interrupt	sys_aon
int_req_apcpu_wdg	13	watchdog in aon sys interrupt	sys_aon
int_req_adi_mst	12	adi master interrupt	sys_aon
int_req_ext_rstb_spcpu	11	chip reset sp interrupt	sys_aon
int_req_ap_wdg_rst_flag	10	watchdog in aon sys interrupt	sys_aon
int_req_sec_wdg_rst_flag	9	watchdog in aon sys interrupt	sys_aon
int_req_busmon_aon_main_mtx_master	8	aon matrix Bus Monitor interrupt	sys_aon
int_req_pwr_down_all	7	power down all sys interrupt	pmu
int_req_pwr_up_all	6	power up all sys interrupt	pmu
int_req_pub_pwr_up	5	pub power up interrupt	pmu
int_req_ap_pwr_up	4	AP power up interrupt	pmu
int_req_apcpu_mode_st	3	apcpu status interrupt	pmu
int_req_ana_int0	2	analog interrupt	io_mux
int_req_ana_int1	1	analog interrupt	io_mux
int_req_ana_int2	0	analog interrupt	io_mux
int_req_apcpu_wdg_flag	NMI	apcpu watchdog flag interrupt	sp_wdg

7.5 System Low power Design

SP STAR MCU Power Design

7.5.1 Power Scheme

SP sys is on the always on power domain, so SP sys will never power off. But there are 3 power states in SP STAR MCU: active,light sleep,deep sleep.

Table 7-3 sp_STAR MCU power states information

Power Status	Modules
Active	STAR MCU Core Active
light sleep	Only STAR MCU Core sleep
deep sleep	SP sys sleep

7.5.2 Boot Information

APCPU has to set SP system clock/reset reg for boot.

STAR MCU Boot related register			
AON PMU APB REG (Base Addr 0x6491_0000)			
Name	Offset	Wr Value	Description
FORCE_DEEP_SLEEP_CFG_0	0x0818	Bit[8]=0	bit[8]= SP_FORCE_DEEP_SLEEP_REG default=1,set/clear Force SP deep sleep; 0: not force; 1: force

APCPU copy STAR MCU code from DDR to STAR MCU RAM, then release STAR MCU core soft reset related register.

NOTE

SP_CORE_SOFT_RST default value is 1 and high active,APCPU has to make sure the SP sys is in reset state during code movement, and then release SP_CORE_SOFT_RST to 0.

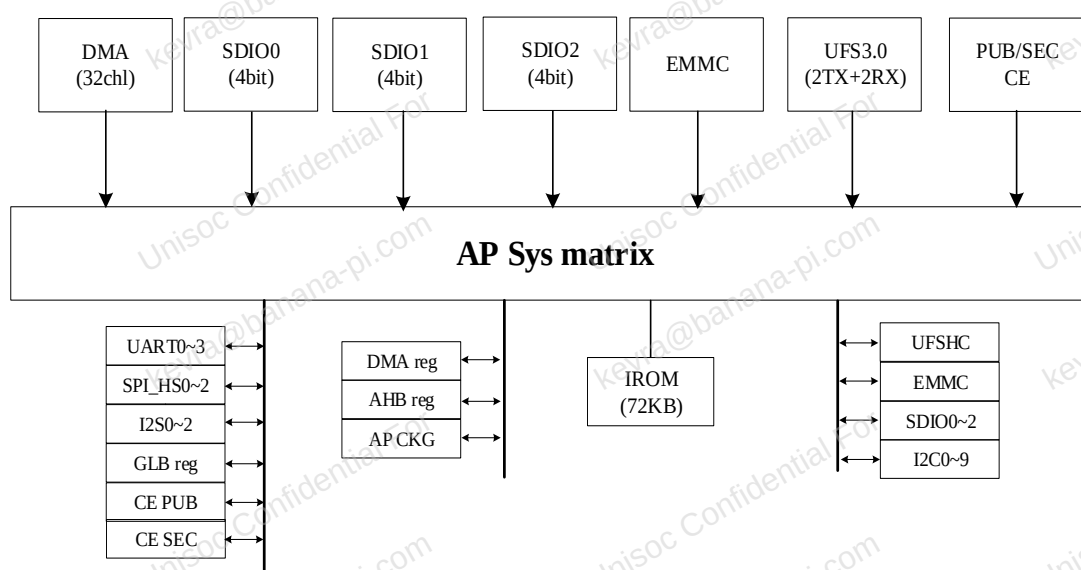
STAR MCU Boot related register			
AON APB REG (Base Addr 0x6490_0000)			
Name	Offset	Wr Value	Description
SP_SYS_SOFT_RST	0x90	Bit[0]=0	bit[0]= SP_CORE_SOFT_RST,default=1,set/clear
		Bit[4]=0	bit[4]= SP_SYS_SOFT_RST,default=0,set/clear

8 AP Subsystem

8.1 Overview

All the masters of application subsystem and ROM are integrated in this system, and the main bus matrix is nic400.

Figure 8-1 Architecture of AP SYS



8.2 Features

- Matrix: one matrix in AP sys, 8 AXI master port, 7 slave port(one APB, one AHB and 5 AXI)
- DMA: Support 64bit axi interface DMA
- Support 3 SDIO: one 4bit SDIO for TF, support SDIO3.0, UHS-1 type, SDR104; one 4bit SDIO for Wi-Fi, support SDR104; one 4bit SDIO for DTV or Security Card, support SDR104
- EMMC: support eMMC5.1 HS400
- UFS: support UFS3.1, 2TX and 2RX
- CE: support AES(ECB, CBC, CTR, GCM, XTS, CMAC, GMAC, Key Length: 128, 192, 256), SHA-1, SHA-256, SHA-512, HMAC, TRNG, RSA(1024, 2048, 3072, 4096), ECC, SM2/SM3/SM4, FIPS 140-2 compliant, NIST800-90 compliant
- Boot ROM: 72KB
- Support 4 UART: one for GNSS, one for **Bluetooth®**, one for log and one backup

- Support 9 I2C: one for sensor, one for GNSS, one for LCD, one for CAM0, one for CAM1, one for CAM2, one for NFC, one for DCDC, one for CTP
- Support 3 SPI: one for sensor, one for FP, one for DTV
- Support 3 IIS: one for Bluetooth, two for codec

8.3 AP Memory Map

The address bus is 35-bit wide and the address map is presented in this section.

Table 8-1 AP memory map

Address Range		Module Name
Start Address	End Address	
0x2010_0000	0x2010_FFFF	sys_ap.apbreg
0x2015_0000	0x2015_FFFF	sys_ap.nic400
0x2020_0000	0x2020_007F	sys_ap.uart0
0x2021_0000	0x2021_007F	sys_ap.uart1
0x2022_0000	0x2022_007F	sys_ap.uart2
0x2023_0000	0x2023_007F	sys_ap.uart3
0x2026_0000	0x2026_007F	sys_ap.spi0
0x2027_0000	0x2027_007F	sys_ap.spi1
0x2028_0000	0x2028_007F	sys_ap.spi2
0x202B_0000	0x202B_007F	sys_ap.iis0
0x202C_0000	0x202C_007F	sys_ap.iis1
0x202D_0000	0x202D_007F	sys_ap.iis2
0x202E_0000	0x202E_1FFF	sys_ap.ce_pub
0x202F_0000	0x202F_1FFF	sys_ap.ce_sec
0x2000_0000	0x2000_FFFF	sys_ap.ahbreg
0x2001_0000	0x2001_FFFF	sys_ap.ckg
0x2002_0000	0x2002_FFFF	sys_ap.dma
0x2100_0000	0x2101_FFFF	sys_ap.rom_ip
0x2224_0000	0x2224_003F	sys_ap.i2c0
0x2225_0000	0x2225_003F	sys_ap.i2c1
0x2226_0000	0x2226_003F	sys_ap.i2c2
0x2227_0000	0x2227_003F	sys_ap.i2c3

Address Range		Module Name
Start Address	End Address	
0x2228_0000	0x2228_003F	sys_ap.i2c4
0x2229_0000	0x2229_003F	sys_ap.i2c5
0x222A_0000	0x222A_003F	sys_ap.i2c6
0x222B_0000	0x222B_003F	sys_ap.i2c7
0x222C_0000	0x222C_003F	sys_ap.i2c8
0x2200_0000	0x221F_FFFF	sys_ap.ufs_hci
0x2220_0000	0x2220_03FF	sys_ap.emmc
0x2221_0000	0x2221_03FF	sys_ap.sdio0
0x2222_0000	0x2222_03FF	sys_ap.sdio1
0x2223_0000	0x2223_03FF	sys_ap.sdio2
0x222D_0000	0x222D_003F	sys_ap.i2c9

8.4 Interrupt

Table 8-2 AP interruption list

INT source name	IR Q	IRQ Trig Type	FI Q	FIQ Trig Type	AP_CPUINT Serial Num	PSCP_CPUINTS erialNum	SP_CPUINTSe rialNum
int_req_slv_fw_ap	Y	H	Y	H	220		
int_req_dma	Y	H	Y	H	219		
int_sec_req_dma	Y	H	Y	H	218		
int_req_emmc	Y	H	Y	H	217		
int_req_i2c0	Y	H	Y	H	216		
int_req_i2c1	Y	H	Y	H	215		
int_req_i2c2	Y	H	Y	H	214		
int_req_i2c3	Y	H	Y	H	213		
int_req_i2c4	Y	H	Y	H	212		
int_req_i2c5	Y	H	Y	H	211		
int_req_i2c6	Y	H	Y	H	210		

INT source name	IR Q	IRQ Trig Type	FI Q	FIQ Trig Type	AP_CPUINT Serial Num	PSCP_CPUINTS erialNum	SP_CPUINTSe rialNum
int_req_i2c7	Y	H	Y	H	209		
int_req_i2c8	Y	H	Y	H	208		
int_req_i2c9	Y	H	Y	H	207		
int_req_iis0	Y	H	Y	H	206		
int_req_iis1	Y	H	Y	H	205		
int_req_iis2	Y	H	Y	H	204		
int_req_sdio0	Y	H	Y	H	203		
int_req_sdio1	Y	H	Y	H	202		
int_req_sdio2	Y	H	Y	H	201		
int_req_spi0	Y	H	Y	H	199		
int_req_spi1	Y	H	Y	H	198		
int_req_spi2	Y	H	Y	H	197		
int_req_ap_uart0	Y	H	Y	H	196		
int_req_ap_uart1	Y	H	Y	H	195		
int_req_ap_uart2	Y	H	Y	H	167		
int_req_ap_uart3	Y	H	Y	H	166		
int_req_ce_pu b	Y	H	Y	H	165		
int_req_ce_se c	Y	H	Y	H	164		
int_req_busm on_ufshc	Y	H	Y	H	163		47
int_req_ufs_fe _int	Y	H	Y	H	162		
int_req_ufs_SI nterrupt	Y	H	Y	H	159		

8.5 AP Clock Control Register

See UIS7885 AP System Control Registers Instructions.

8.6 AP AHB Control Register

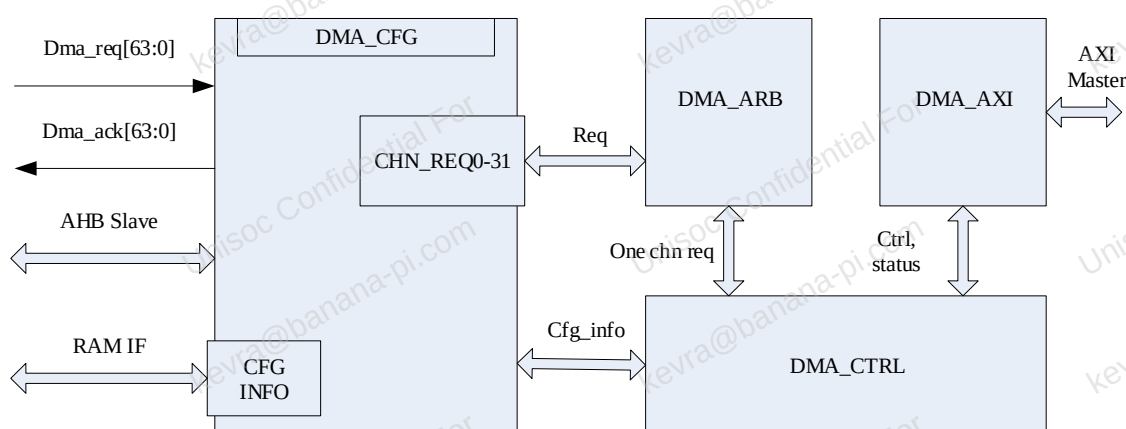
See UIS7885 AP System Control Registers Instructions.

8.7 AP APB Control Register

See UIS7885 AP System Control Registers Instructions.

8.8 DMA

8.8.1 Overview



8.8.2 Features

- Arm® AMBA® 3 AXI Specification.
- Two 64-bit AXI masters for read and write operation.
- One AHB 32-bit slave for configuration only supports word access to AHB slave port.
- Support Byte/Half-word/word transfer on AXI Read/Write Master.
- Not support Arm® AMBA® EXOKAY/SLVERR/DECERR response.
- Support Arm® AMBA® INCR transaction, burst length is calculated by transaction Byte length.
- Support max outstanding number is 16.
- Support 64GB address space.
- AXI Master Big-endian or Little-endian.
- Support 32 channels assigned to hardware request by Channel ID;
- One physical channel shared by 32 Logic channels.
- 4-level configurable priority. 3 has the highest and 0 has the lowest priority. For the channels with the same priority, round-robin would be preferred.
- A-synchronized/Synchronized hand-shake request & acknowledge.
- Accept request from both hardware and software.
- Un-aligned word accesses, for both read and write operation.

- Byte/Half-word switch scheme.
- Support link list for scatter and gather transfer.
- Support one channel to start the other channel.
- Configurable fragment wait time between two successive fragments.
- One request for fragment/block/transaction/link-list.
- Automatically disable channel after requests done.
- Support (128K-1) Bytes fragment length.
- Support (128K-1) Bytes block length (Block length may be less than fragment length in Standard Channel, while forbidden in Full channel).
- Support (256M-1) Bytes transaction length.
- Support transfer step, ranged from – (32K-1) Bytes to (32K-1) Bytes, only for full DMA channel.
- Support fragment step, ranged from – (32K-1) Bytes to (32K-1) Bytes, only for full DMA channel.
- Support block step, ranged from – (128M-1) Bytes to (128M-1) Bytes, only for full DMA channel.
- Address-fix transfer for either Source side or destination side.
- Address-wrap transfer for either Source side or destination side.
- Pause DMA or pause logical channel.
- Interrupt of fragment, block, transaction and link list, with raw and masked status.
- Interrupt of Error configuration, and support errors status.
- Support hardware request to be re-allocated by Channel ID.
- Support automatically sleep when all requests served, and wakeup when hardware request active.
- Support software retention if power domain of DMA shut down.

8.8.3 Function Description

8.8.3.1 General Description

DMA Controller support fast data transfers to off-load the processor. With DMA, specific devices on AXI and APB can complete quick data movement from/to memories, internal RAM or External SRAM. DMA Controller can also be used to connect any two devices other than memories, as long as they can be addressed in memory space.

8.8.3.2 Standard & Full DMA Channels

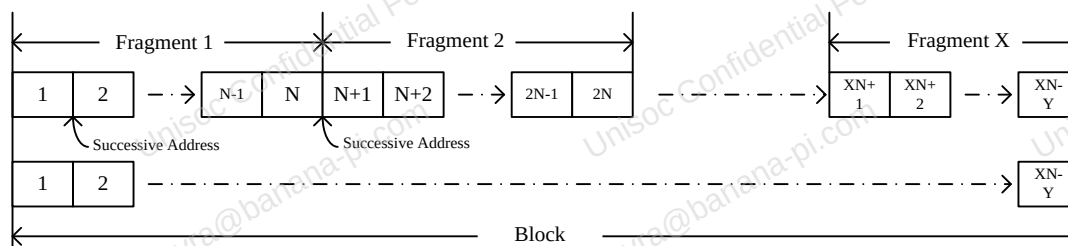
There are two types of DMA channels. One is called “Standard” DMA channel, and the other is “Full” DMA channel.

For most applications, a standard DMA channel would be sufficient, with least configurations and simplest control flow. AXI sequential transfer and FIFO transfer would be efficiently managed, with Byte, half-word or word size.

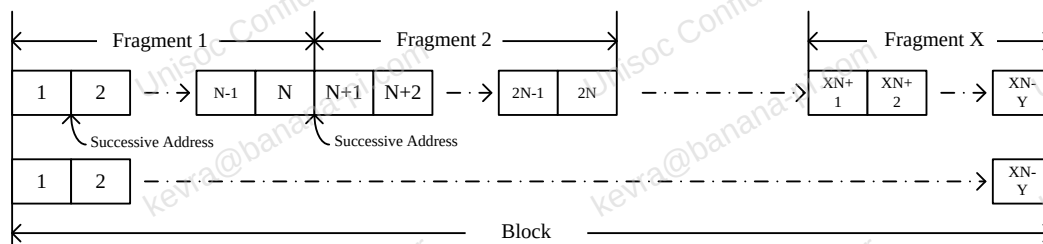
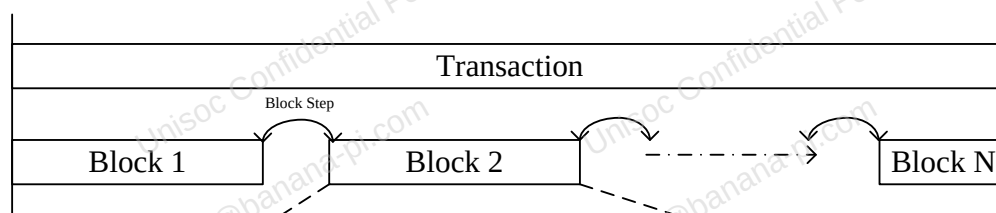
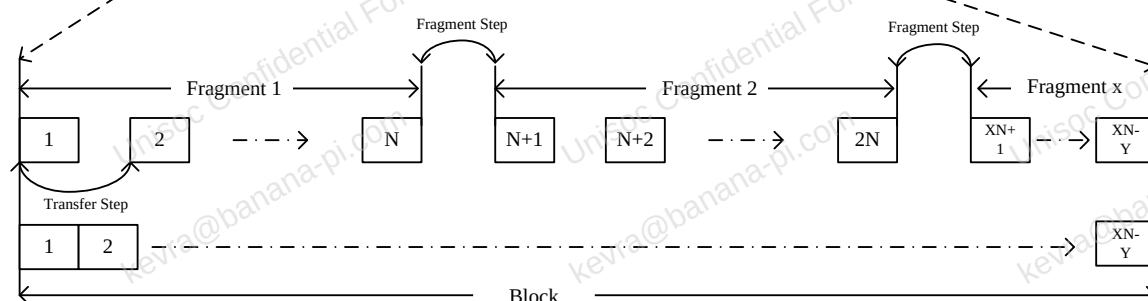
For special applications, such as image rotation or complex data arrangement, Full DMA channel support address wrap transfer, configurable transfer step and link list mode.

8.8.3.3 Data Structure

- For Standard DMA channel



- For Full DMA channel



8.8.3.4 DMA Pause

There are two levels of pause action implemented in DMA controller.

- One is “controller pause” suitable for the whole DMA controller. Once the controller pause occurred, the DMA controller would stop all the channels after the current fragment/block completed, hold the accepted and subsequently requests. DMA would not sleep during pause state.
- One is “channel pause” suitable for the single DMA channel. When the channel pause occurred, the channel would be temporarily hold and stop all the transfer action.

8.8.3.5 Channel Priority and Arbitration

Each channel may be configured to be one of total 4 level priorities, to effectively manage the bandwidth among all logical channels. If there is more than one request occurring, DMA controller would firstly judge their priorities and then select the highest one to serve.

For the channels having the same level priority, DMA controller would serve them by Round-robin way.

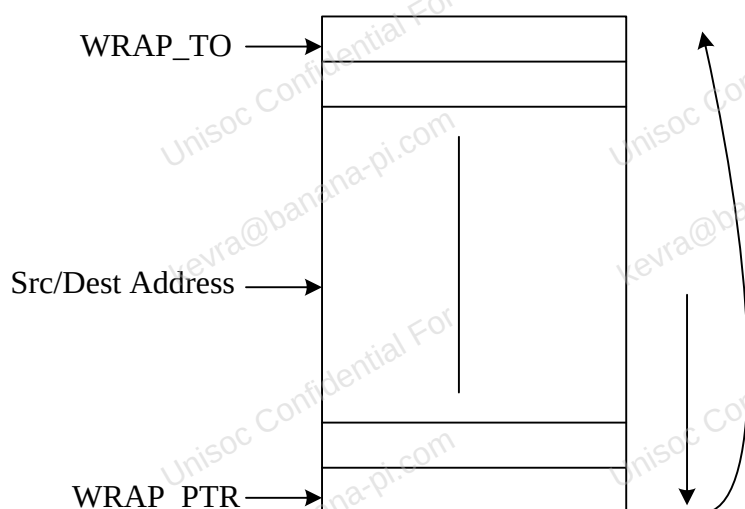
8.8.3.6 Address Fix Mode

Both Standard and Full DMA channel support address-fix transfer, on either source or destination side. This mode is optimized for UART/SPI etc peripherals, and fragment length “FRAG_LEN” would be configured equal to the FIFO watermark.

8.8.3.7 Address Wrap Mode

Full-DMA channel support data movement of ring-buffer and double-buffer, as illustrated below. Once the address reaches the “WPPT”, the next address jumps to the “WPTO” address after completing the “WPPT” data transfer.

Attention the data at “WPPT” isn’t moved to the destination address.



8.8.3.8 Endian & Data Switch

DMA controller would be configured to be big-endian or little-endian on AXI master ports. The detailed information on AXI big-endian and little-endian would be preferred to AMBA® 3 Specifications.

8.8.3.9 Un-aligned Word Access

Word access on AXI bus support burst type of address un-aligned to word boundary. If DMA was configured to read/write un-aligned address with word size, when DMA read, the DMA would start axi burst to read memory by the un-aligned start address, then the DMA discard redundancy Byte and selects valid data, when DMA write, the DMA would start un-aligned burst transfer, but the valid data information must be in the strobe signal.

DMA AXI burst types are all INCR, the burst length is 1-16.

DMA probably had to read/write them by Byte, as illustrated below (The red Byte is redundancy Byte).

Start address:0x00
Transfer size:32bit
Burst type:INCR
Burst length:5 transfers

3	2	1	0	1st transfer
7	6	5	4	2nd transfer
11	10	9	8	3rd transfer
15	14	13	12	4th transfer
19	18	17	16	5th transfer

Start address:0x01
Transfer size:32bit
Burst type:INCR
Burst length:5 transfers

3	2	1	0	1st transfer
7	6	5	4	2nd transfer
11	10	9	8	3rd transfer
15	14	13	12	4th transfer
19	18	17	16	5th transfer

Start address:0x02
Transfer size:32bit
Burst type:INCR
Burst length:5 transfers

3	2	1	0	1st transfer
7	6	5	4	2nd transfer
11	10	9	8	3rd transfer
15	14	13	12	4th transfer
19	18	17	16	5th transfer

Start address:0x03
Transfer size:32bit
Burst type:INCR
Burst length:5 transfers

3	2	1	0	1st transfer
7	6	5	4	2nd transfer
11	10	9	8	3rd transfer
15	14	13	12	4th transfer
19	18	17	16	5th transfer

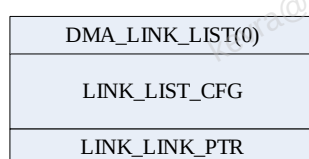
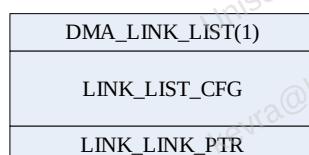
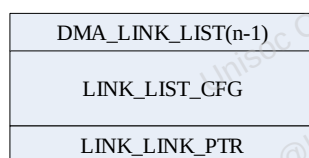
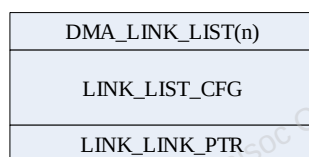
For the memory space with un-aligned word access, Byte/Half-word/Word size access must be supported.

For Full DMA channel, if the step of transfer different with the data size (the address is not successive), un-aligned word access would turn to be total Byte access.

8.8.3.10 Word Packing

To maximum utilizing internal FIFO, and reduce AXI wait for un-aligned word access, Byte or half-word would be packed into word internal DMA.

8.8.3.11 Link List



Software can create a few link list, and then connect these list, the link_list_end flag of the last list should be high means list sequence is end.

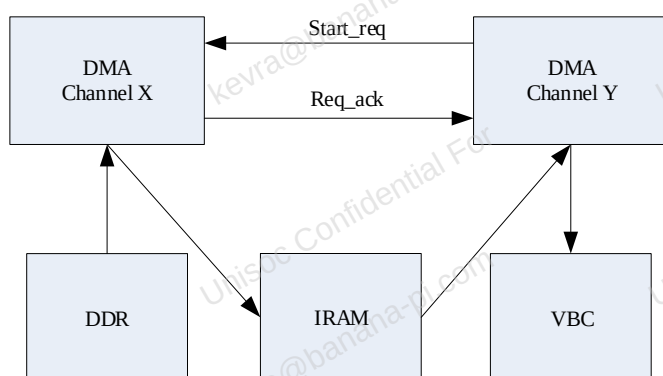
8.8.3.12 Hardware Request Re-allocated

Every device (peripherals or accelerators) would be assigned to a hardware request, when DMA was implemented in SOC. DMA controller support each hardware request to be re-allocated to the Standard or Full DMA channels.

8.8.3.13 One Channel Start Other Channel

This Function is optimized for audio, firstly, the software must configure the only two channel registers, and ensure the two channels can't be used by any other hardware request or software request, and each channel can start the other channel.

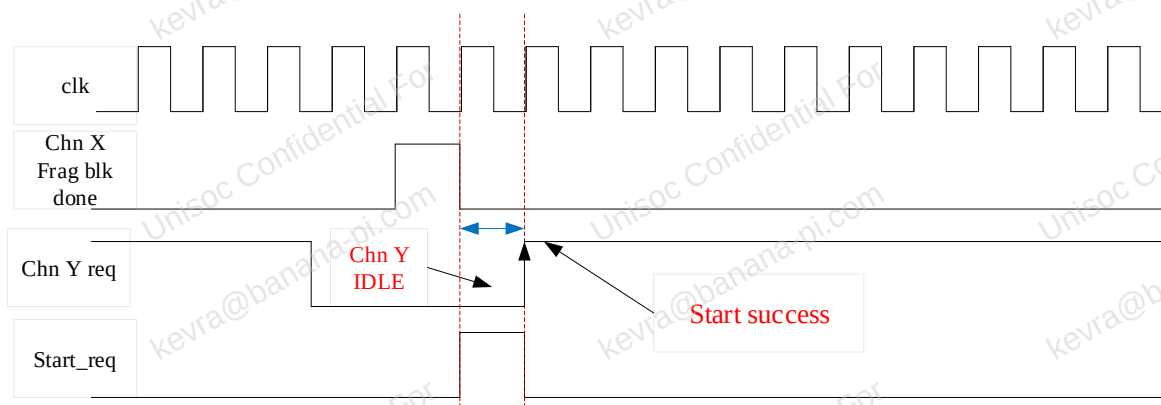
DMA audio application as illustrated below.



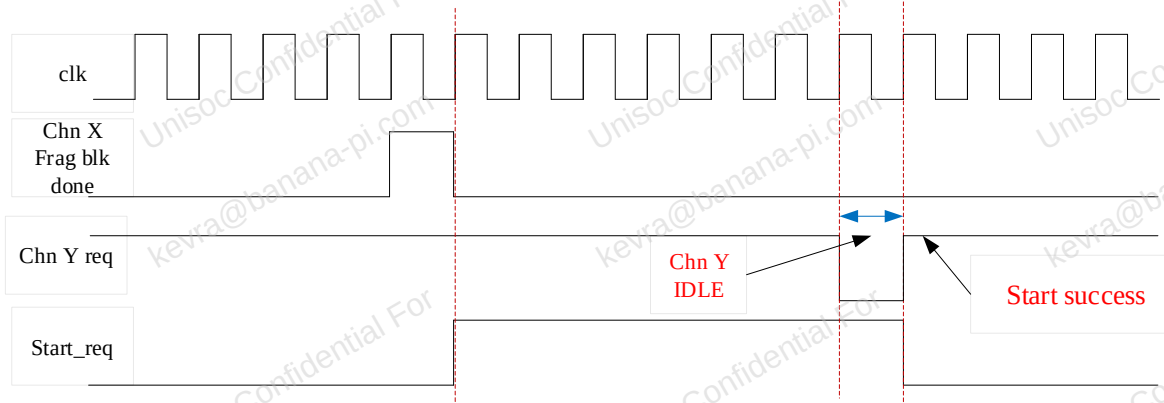
Firstly, CPU should start channel X only one time, when the IIRAM has enough data from channel X, CPU (or VBC) can start channel Y to shift data from IIRAM to VBC. when channel X is idle and a small amount data in IIRAM, channel Y can start channel X when channel Y frag_done or blk_done or trsc_done or llist_done.

Start control signal timing:

- The channel Y is idle state when channel X start request is valid, then next clock cycle start channel Y is successful.



- The channel Y is busy state when channel X start request is valid, so the channel X request will hold until channel Y is idle, when the channel Y is idle, then next clock cycle start channel Y is successful.



8.8.3.14 Error Configuration

1. Unaligned word/halfword access.
2. Request for transaction or link list in Standard DMA channel.
3. Zero Fragment/block/transaction length.
4. Address wrap for Standard Channel.

8.8.4 Control Registers

See *UIS7885 AP System Control Registers Instructions*.

8.8.5 Application Notes

- In Standard Channel, “frag_len” would be larger or smaller or equal to “blk_len”, and “blk_len” would be updated after each “frag_len” completed. “blk_cnt” would be no longer exists. “blk_len_rec_h/l” not available in Standard Channel.
- In Full Channel, “Frag_len”, “blk_len” and “trsc_len” would be any relationship of size. “blk_len” would not be updated after “frag_len” completed and “blk_len_rec_h/l” would be updated.
- In Standard Channel, not support 8GB address space
- In Full Channel, support 64GB address space, but the llist_ptr can’t cross the 4GB boundary.
- Dma_64b_top uses 64bits axi bus.
- When src/dst data size equal 2'b01, the all addr/step/length/wrap addr to/ptr should be half-word align.
- When src/dst data size equal 2'b10, the all addr/step/length/wrap addr to/ptr should be word align.
- When swt_mode != 2'b00, or data format is big-endian, the all address and length should be word align.
- Link list pointer should be DWORD align.

8.9 SDIO0/SDIO1/SDIO2

Same as to eMMC, please refer section [8.10 eMMC Host](#).

8.10 eMMC Host

8.10.1 Overview

The eMMC Host Controller is a Host Controller with an Arm® processor interface. The eMMC Host Controller handles eMMC/eMMC Protocol at transmission level, packing data, adding cyclic redundancy check (CRC), start/end bit, and checking for transaction format correctness.

8.10.2 Features

For the practicable reason, the used and unused features are list below, the others based on the standard specifications which not list are supported

- Meet SD Host Controller Standard Specification Version 4.10
- Meet SDIO card specification version 3.0
- Meet JEDEC eMMC spec JESD84-B51
- Capacity of Memory
 - (SDSC) (2G) Standard Capacity SD Memory Card
 - (SDHC) (32G) High Capacity SD Memory Card
 - (SDXC) (2T) Extended Capacity SD Memory Card
- Bus Speed Mode (SD Card)
 - Default Speed: Frequency <= 25MHz, Data rate<= 12.5MB/s
 - High Speed: Frequency <= 50MHz, Data rate<= 25MB/s
 - SDR12:UHS-I Frequency <= 25MHz, Data rate<= 12.5MB/s
 - SDR25:UHS-I Frequency <= 50MHz, Data rate<= 25MB/s
 - SDR50:UHS-I Frequency <= 100MHz, Data rate<= 50MB/s
 - SDR104:UHS-I Frequency <= 208MHz, Data rate<= 104MB/s
 - DDR50: UHS-I Frequency <= 50MHz, Data rate<= 50MB/s
- Bus Speed Mode (eMMC Card)
 - Legacy MMC: Frequency <= 26MHz, Data rate <= 26MB/s
 - High Speed SDR: Frequency <= 52MHz, Data rate <= 52MB/s
 - High Speed DDR: Frequency <= 52MHz, Data rate<= 104MB/s
 - HS200: Frequency <= 200 MHz, Data rate<= 200MB/s
 - HS400: Frequency <= 200 MHz, Data rate<= 400MB/s
- Support SDMA/ADMA2/ADMA3 mode of operation (AXI interface)
- Support 64bit/32bit address descriptor
- Support 1 bit, 4 bit ,8bit SD modes
- Support card to interrupt host in 1bit, 4 bit SD modes.
- Support CRC7 for command and CRC16 for data
- Support Read waits Control.
- Support Stop Clock for read busy
- Support CMD0 with Argument 32'hFFFFFFFA or CMD line low to boot eMMC
- Support battery remove and card remove protect function
- Support Byte Configuration for software interface APB bus

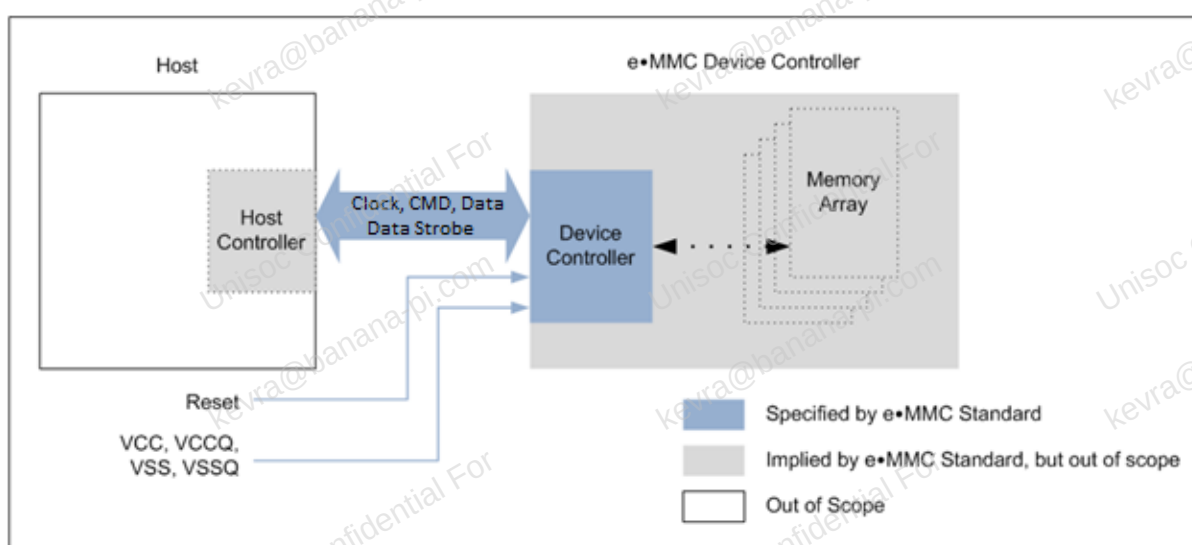
- Not support SDMA buffer boundary
- Not support No-DMA mode
- Not support Stop at block gap
- Not support Suspend & Resume
- Not support Infinite mode

8.10.3 Function Description

8.10.3.1 eMMC System Overview

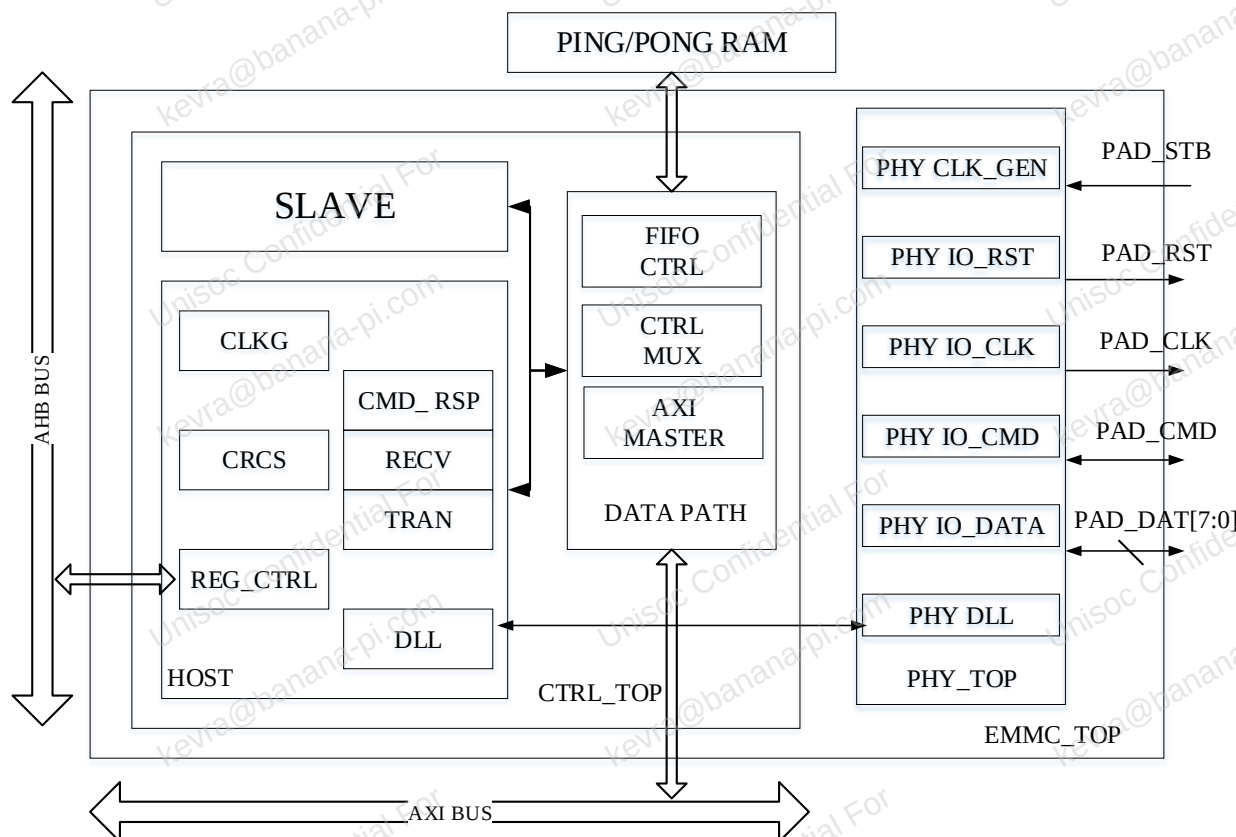
The connection of eMMC host PADs is shown in the following figure.

Figure 8-2 eMMC System Overview



8.10.3.2 Controller Block Diagram

Figure 8-3 eMMC host System Block Diagram



The EMMC_TOP module includes the CTRL_TOP and PHY_TOP. The CTRL_TOP is the digital part which is writes in RTL code, and the PHY is described in standard cells which can not be synthesis again.

- The PHY_TOP part ensures the output or sample timing close meet the high demand. The common require for the PHY is not to insert buffers, and the placement should meet the guide.
- The CTRL_TOP include 3 parts of HOST, DATA_PATH, and SLAVE.
 - The HOST part is the main part of the eMMC Host control, includes REG_CTRL, CLKG, CMD_RSP, TRAN, RECV, CRCS, and DLL.
 - The REG_CTRL interface acts as the bridge between the Arm® processor and Host Controller. The eMMC controller registers are programmed by the Arm® processor through AHB target interface. Interrupts are generated to the Arm® processor based on the values set in the Interrupt status register and Interrupt enable registers.
 - The CLKG Clock generation block will generate the SD clock depending on the value programmed by the Arm® processor in the Clock Control Register.
 - The CMD_RSP send command (such as CMD0, CMD1, CMD18) to eMMC or SD card, and received response from card.
 - TRAN module transmits data to eMMC card and RECV module receives the data from eMMC card.

- The CRCS includes CRC7 and CRC16 which generator calculates the CRC for command and Data respectively to send the CRC to the eMMC card. The CRC7 and CRC16 checker checks for any CRC error in the Response and Data send by the eMMC card.
- DLL is newly added module which is used to track the clock frequency of the working clock, which should co-work with eh PHY_DLL.
- The DATA_PATH include AXIM and BUF_CTRL.
 - The AXI master (AXIM) initiates a read or write transaction with the memory if the Data transaction is done using DMA data transfer method.
 - The BUF_CTRL used a ping-pong buffer to generate the read-write enable, full, empty Signals. Which give the control the Signals to generate the read busy, write wait control Signals.
- The SLAVE is another design which just shared the DATA_PATH with the HOST.

8.10.3.3 PHY Block Logic and Timing

The PHY block control the data input and output timing. Following is the brief timing relation of the CLK/CMD/DAT.

Figure 8-4 eMMC PHY CLK/CMD/DAT Timing Relation

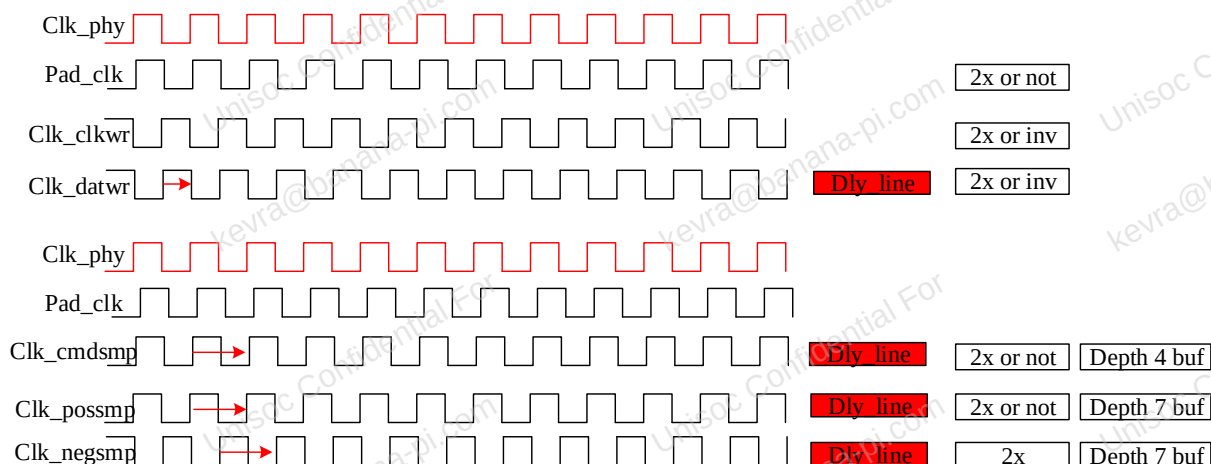
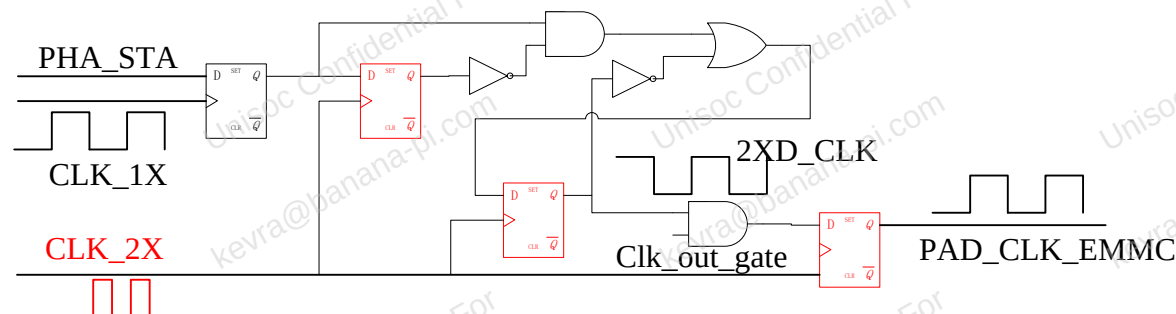


Figure 8-5 eMMC PHY CLK Logic



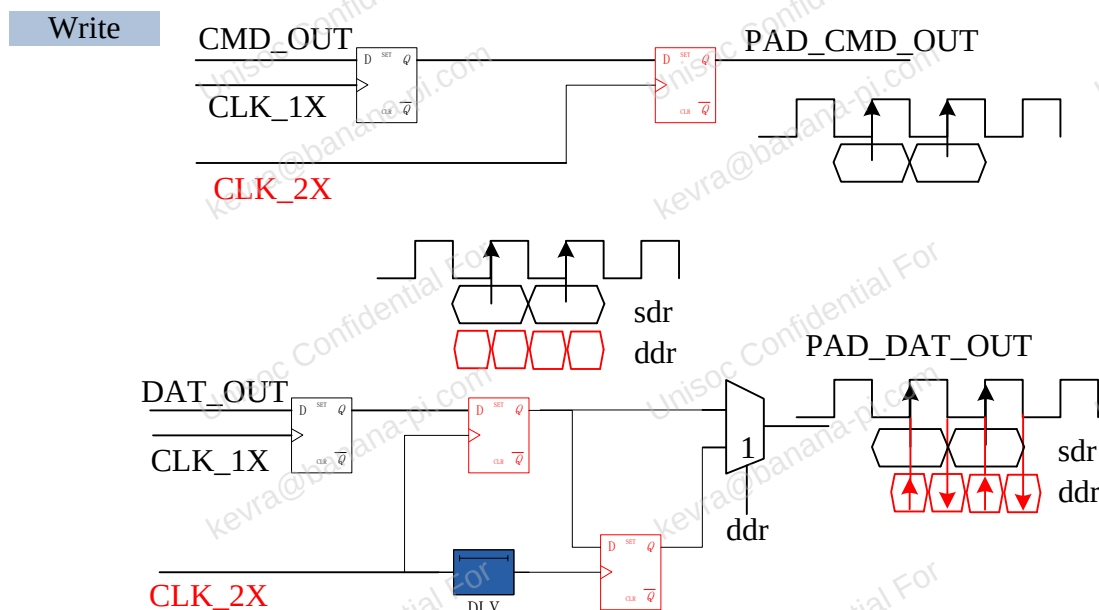
The CLK output is directly clk_emmc_1x clock or generated from clk_emmc_2x by divider 2 with same phase as clk_emmc_1x based on the timing mode. When high speed mode, use 2x clock divider, else used 1x clock.

- When Write.

The CMD output is invert clock phase with clock that means the CMD is always negedge clock output. The last step clock which registered the CMD may be invert of clk_emmc_1x or clk_emmc_2x divide by 2 with invert phase with clk_emmc_1x base on the timing mode. When high speed mode, use 2x clock divider, else used 1x clock.

The 1st step of DAT output are registered by 2x clock or inv clock, the 2nd step has delay line to generate ¼ cycle delay. When high speed mode, the final DAT output is the 2nd step data output, else the final DAT output is the 1st step data output.

Figure 8-6 eMMC PHY CLK, CMD/DAT Write Logic



- When Read.

The 1st step of CMD input is the delayed clock of clk_emmc_1x, which can be delayed by one clock cycle, then the data is stored in a depth 4 buffers; the 2nd step of CMD input is registered by clk_emmc_1x clock which is synchronous with the internal processing clock. When in not low speed, the 1st step of CMD need not to store buffer, and just with some little delay, and the 2nd step of clock directly register it.

The DAT input is same as CMD with 2 steps sample clock. When in HS400 mode, the first step sample clock is used the input STOB; when in others mode, the first step sample clock used 2x divider clock or directly 1x clock based on the timing mode. When in no DDR mode, the negedge sample clock is not working.

Read

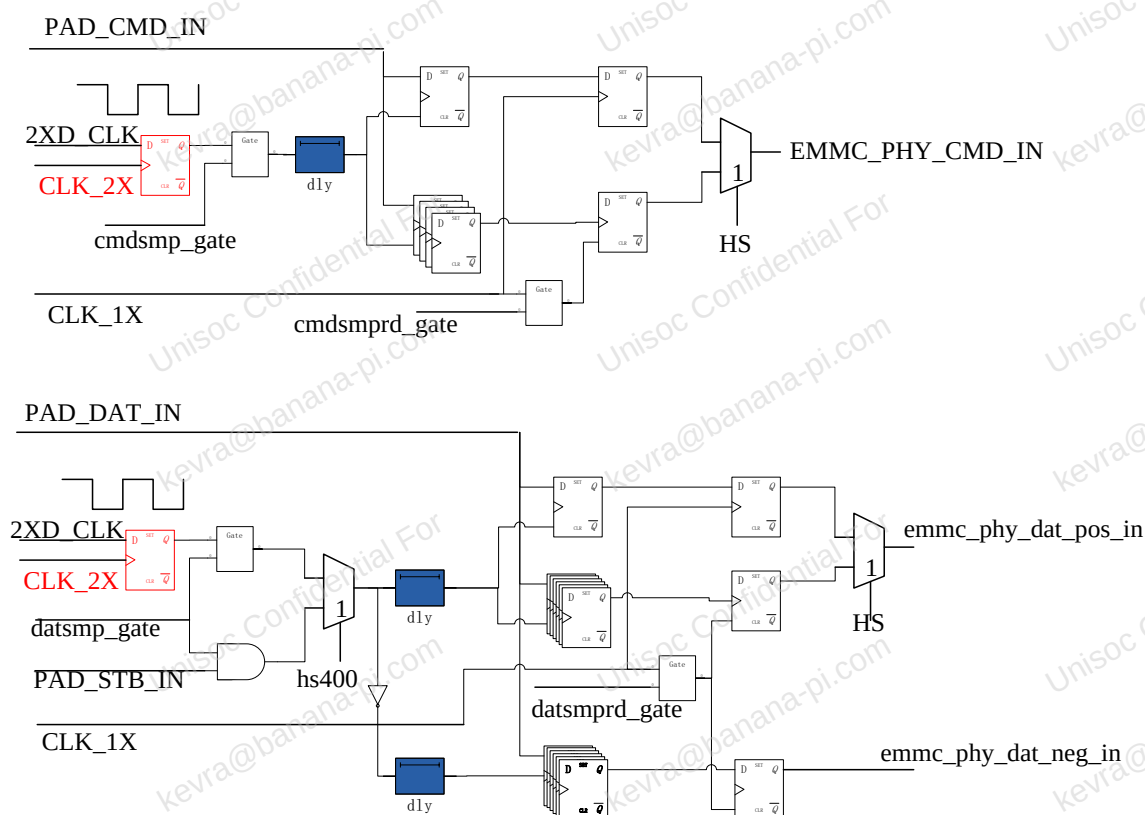


Table 8-3 Signals List

Table 8-3 Signals List

Pin Name	I/O	Width	Description
Bus Interface			
APB Slave Signals			Standard APB Slave signal
AXI Mast Signals			Standard AXI Master signal
RAM interface			
RAM Ping Signals			64 (width) × 128 (depth)
RAM Pong Signals			64 (width) × 128 (depth)
System Interface			
Ptest_icg_mode	I	1	eMMC clock gate test mode select
Ptest_scan_mode	I	1	eMMC scan test mode select
Ptest_bist_mode	I	1	eMMC internal sram BIST mode select

Pin Name	I/O	Width	Description
Ptest_bist_clkssel	I	1	eMMC clock select in BIST mode
Ptest_scan_clkssel	I	1	eMMC clock select in scan mode
Clk_emmc_host	I	1	eMMC host clock signal from clk_top
Rst_emmc_host_n	I	1	eMMC host reset signal from rst_top
Rst_emmc_por_n	I	1	eMMC host power on reset signal from rst_top
Emmc_soft_rst	I	1	eMMC host soft reset signal from AP/TOP
Emmc_freq_div	O	10	eMMC clock divider to clk_top
Int_req_emmc	O	1	eMMC interrupt signal
PHY interface			
emmc_xxx(excluding Emmc_freq_div[9:0]/Emmc_soft_rst)			Internal Signals between PHY & CTRL
Oe_ext_optional	O	1	Internal signal to PHY
Card protect interface			
Card_det	I	1	Card detect signal
Battery_det	I	1	Battery detect signal
Clk_32k	I	1	32k clock
Rst_32k_n	I	1	Resert_n in 32k domain
Power_off	O	1	Power off signal to adi
Power_off_delay	O	1	Power off signal to adi

8.10.3.5 PHY_TOP Signals

Table 8-4 Signals List

Pin Name	I/O	Width	Description
System Interface			
Ptest_icg_mode	I	1	eMMC clock gate test mode select
Ptest_sacn_mode	I	1	eMMC scan test mode select
Clk_emmc_host	I	1	eMMC host clock signal from clk_top
Clk_emmc_host_2x			eMMC host clock 2x signal from clk_top
Rst_emmc_por_n	I	1	eMMC host reset from top
PAD interface			
Pad_oe_emmc_stb	O	1	PAD Strobe Signals

Pin Name	I/O	Width	Description
Pad_ie_emmc_stb	O	1	PAD Strobe Signals
Pad_in_emmc_stb	I	1	PAD Strobe Signals
Pad_oe_emmc_rst	O	1	PAD Reset Signals
Pad_ie_emmc_rst	O	1	PAD Reset Signals
Pad_in_emmc_rst	O	1	PAD Reset Signals
Pad_out_emmc_clk	O	1	PAD Clock Signals
Pad_oe_emmc_clk	O	1	PAD Clock Signals
Pad_ie_emmc_clk	O	1	PAD Clock Signals
Pad_in_emmc_clk	I	1	PAD Clock Signals
Pad_out_emmc_cmd	O	1	PAD Command Signals
Pad_oe_emmc_cmd	O	1	PAD Command Signals
Pad_ie_emmc_cmd	O	1	PAD Command Signals
Pad_in_emmc_cmd	I	1	PAD Command Signals
Pad_out_emmc_data	O	8	PAD Data Signals
Pad_oe_emmc_data	O	8	PAD Data Signals
Pad_ie_emmc_data	O	8	PAD Data Signals
Pad_in_emmc_data	I	8	PAD Data Signals
IOMUX interface			
Iomux_emmcstb_sel	I	1	IOMUX Strobe Signals
Iomux_emmcstb_ie	I	1	IOMUX Strobe Signals
Iomux_emmcstb_oe	I	1	IOMUX Strobe Signals
Iomux_emmcstb_in	O	1	IOMUX Strobe Signals
Iomux_emmcrst_sel	I	1	IOMUX Reset Signals
Iomux_emmcrst_ie	I	1	IOMUX Reset Signals
Iomux_emmcrst_oe	I	1	IOMUX Reset Signals
Iomux_emmcrst_out	I	1	IOMUX Reset Signals
Iomux_emmcclk_sel	I	1	IOMUX Clock Signals
Iomux_emmcclk_in	O	1	IOMUX Clock Signals
Iomux_emmcclk_ie	I	1	IOMUX Clock Signals
Iomux_emmcclk_oe	I	1	IOMUX Clock Signals
Iomux_emmcclk_out	I	1	IOMUX Clock Signals

Pin Name	I/O	Width	Description
Iomux_emmccmd_sel	I	1	IOMUX Command Signals
Iomux_emmccmd_in	O	1	IOMUX Command Signals
Iomux_emmccmd_ie	I	1	IOMUX Command Signals
Iomux_emmccmd_oe	I	1	IOMUX Command Signals
Iomux_emmccmd_out	I	1	IOMUX Command Signals
Iomux_emmccdata_sel	I	8	IOMUX Data Signals
Iomux_emmccdata_in	O	8	IOMUX Data Signals
Iomux_emmccdata_ie	I	8	IOMUX Data Signals
Iomux_emmccdata_oe	I	8	IOMUX Data Signals
Iomux_emmccdata_out	I	8	IOMUX Data Signals
PHY interface			
emmc_xxx			Internal Signals between PHY & CTRL
Oe_ext_optional	I	1	Internal signal From CTRL

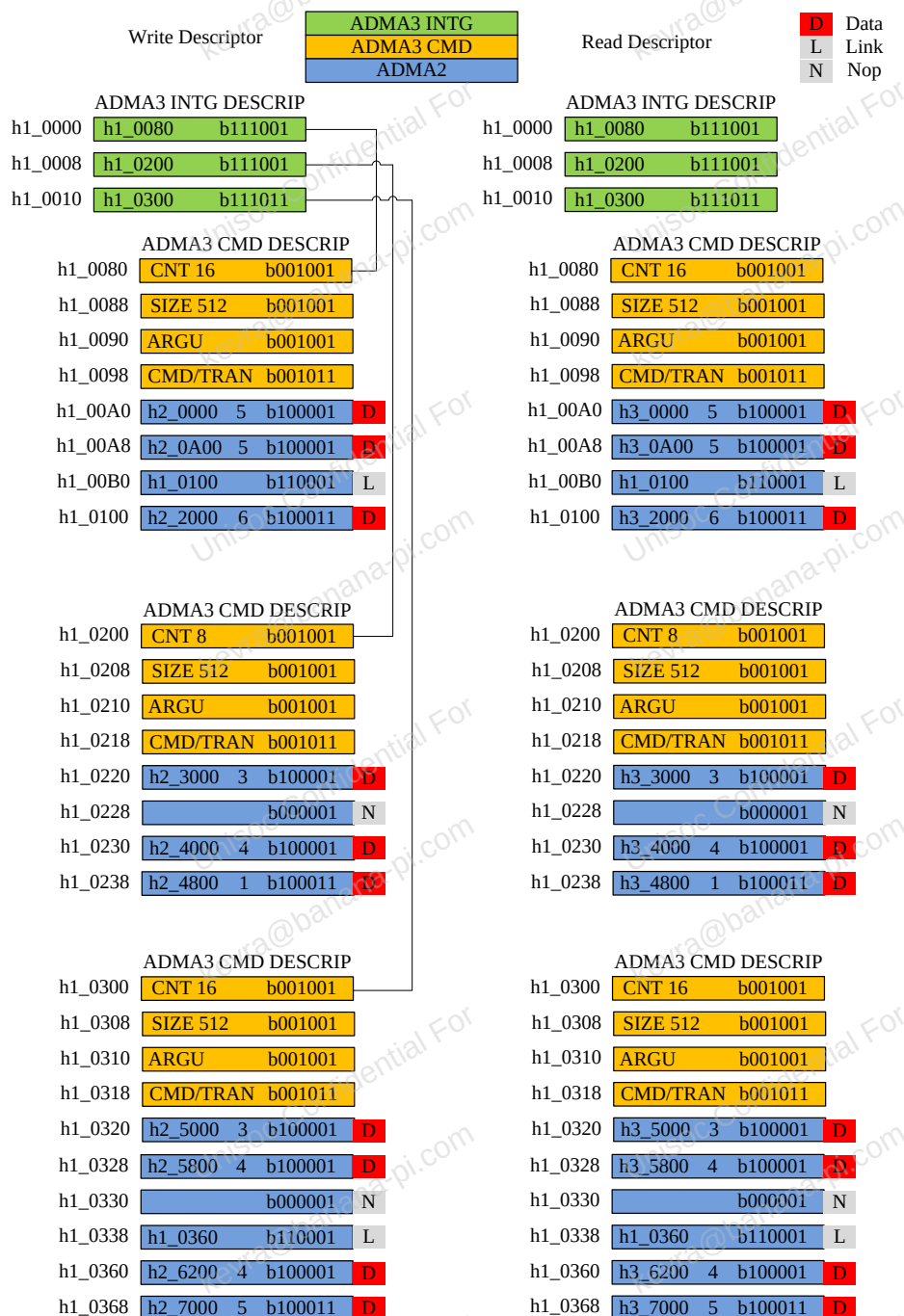
8.10.4 Control Registers

See UIS7885 AP System Control Registers Instructions.

8.10.5 Application Notes

8.10.5.1 ADMA3 example

Figure 8-8 ADMA3 example

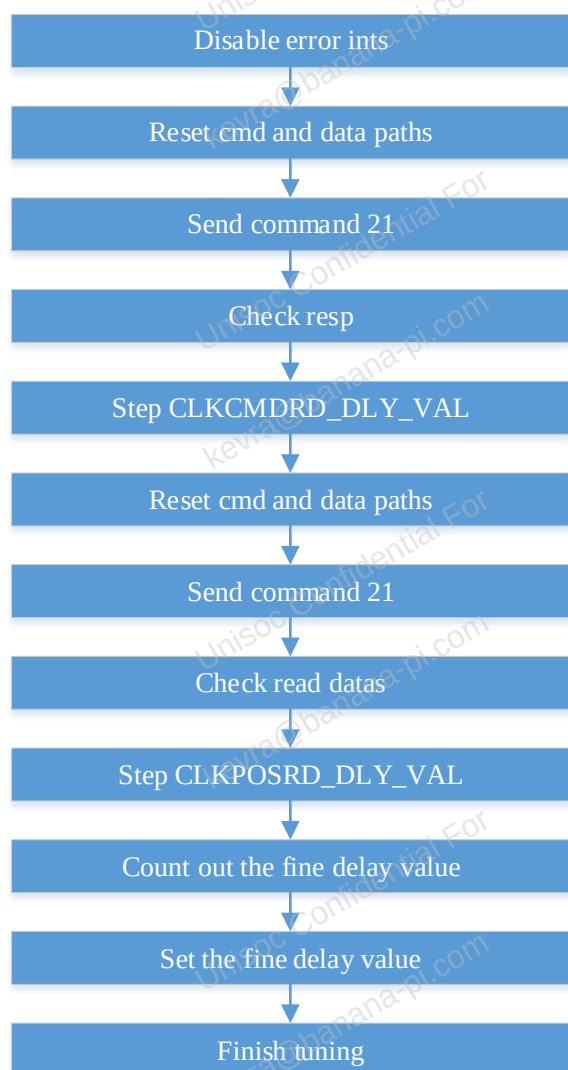


8.10.5.2 Tuning sequence

In HS200 mode, set CPST_EN=0, DLL_CLK_SEL=0

Scan range from 0 to $1.5 \times \text{dll_cnt}$, get the larger pass window that is from x_1 to x_2 ;

Figure 8-9 tuning sequence

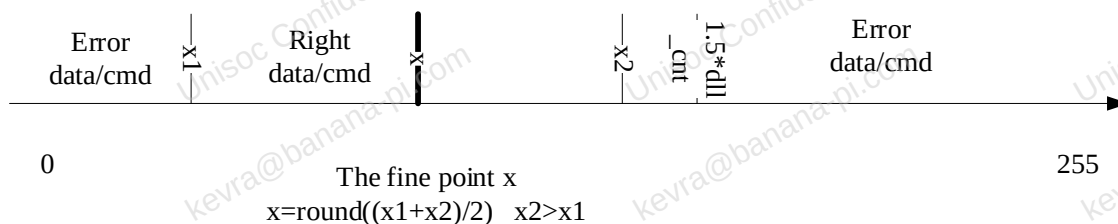


8.10.5.3 Timing select

The middle point of pass window is $X = (x_1 + x_2)/2$;

When you set $\text{CPST_EN}=1$, change the delay value to $(X/\text{dll_cnt}) \times 0\text{xFF}$ if $X \leq \text{dll_cnt}$; 0xFF if $X > \text{dll_cnt}$.

Figure 8-10 the fine tuning point



8.10.5.4 Timing adjust

Mode	Speed	Write delay	Read delay	Command delay
SDR	50MHz	Don't touch	scan	scan
	100MHz	Don't touch	scan	scan
	200MHz	Don't touch	tuning	scan
DDR	50MHz	0x7F	scan	scan
	200MHz	0x3F	0x3F	Hs200 mode scan value

8.11 UART Module

8.11.1 Overview

UART is an asynchronous communication interface. The UART module includes a baudrate generator with software-programmable divider ratios for all common baudrates.

Two types deep FIFO are include. Receive FIFO which is 128-Byte-deep and send FIFO which can be configured to 128-Byte-deep or 256-Byte-deep;The module also includes a flexible interrupt with multiple maskable interrupt sources. Two hardware flow control lines are included (one for input, the other for output). This module also has a built-in IrDA controller and can be configured to connect to an IrDA transceiver. When being used as regular UART port, the IrDA controller is bypassed.

8.11.2 Features

- Full-duplex operation
- Hardware flow control support
- Software flow control support with configurable command and escape register
- Configure TX FIFOs minimizes 128-Byte-deep or 256-Byte-deep processor overhead at high data rates
- Wide selection of UART word lengths, including 5, 6, 7 and 8 bits
- Stop bit number can be 1, 1.5 or 2 bits
- RX stop bit number and TX stop bit number can be configured separately
- Automatic baudrate detection flow support
- Support odd/even parity
- Auto detect for parity and framing error

- UART loop-back test mode
- Break character detection and generation
- support DMA operation
- Single interrupt line for multiple maskable interrupt source events
- Programmable interrupt trigger levels for FIFOs
- Baudrate generation based upon programmable divisors, operating from a flexible functional clock
- Includes a built-in IrDA controller
- Support 32 bits or 8 bits write once to tx FIFO

8.11.3 Function Description

The UART and IRDA baudrate generator is based on the `clk_uart`, the divisor coefficient is controlled by register `UART_CKD0`. IRDA part clock period is the 1/16 of the UART part since the required timing of IRDA is 3/16 or 4/16 of bit width. The serial data rate can be changed by modify the `clk_uart` or the UART divisor coefficient.

Figure 8-11 Diagram of UART

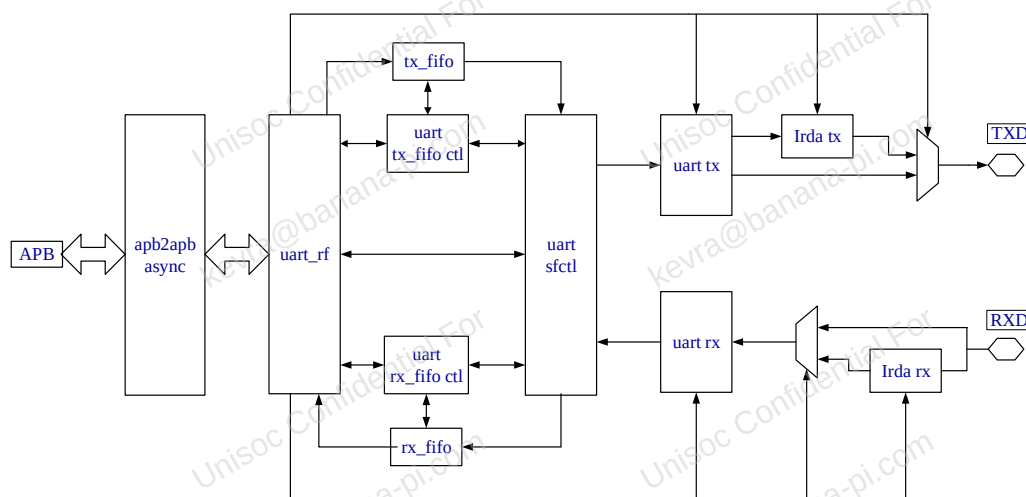


Figure 8-11 is the architecture diagram of the UART module, as it depicted in the graphic, the UART module has separated RX and TX channel to realize the duplex working mode. And the UART module also supports to enable the IRDA transceiver mode.

8.11.3.1 UART Mode Description

UART module has an independent baudrate generator, which is used as a reference clock for data transmit and recovery. It also has a 128-Byte-deep RX FIFO and a 128-Byte-deep TX FIFO to decrease processor interrupt load during the data receiving and transmitting.

8.11.3.2 FIFO Management

FIFOs are accessed by reading/writing `UART_RXD/UART_TXD` register. The number of data in TX/RX FIFO can be observed by reading `UART_STS1 [6:0]/UART_STS1 [15:8]`. Rx_fifo_full and Tx_fifo_empty status can be obtained by reading `UART_STS0 [1:0]`. `UART_CTL2` register controls the FIFO trigger level, which enables the DMA request and interrupts generation.

- TX FIFO size is 128 Bytes or 256 Bytes. It uses a FIFO pointer to monitor the number of data in the TX FIFO that has not yet been transmitted. The MCU can read the UART_STS1 [15:8] to access the pointer. The UART also provides a TXF empty interrupt threshold that can be configured by writing UART_CTL2 [15:8].
- RX FIFO size is 128 Bytes. It uses a pointer to monitor the number of data in the RX FIFO that have not yet been read by the MCU. The MCU can access the pointer by reading the UART_STS1 [6:0]. The UART also provides a RX FIFO full interrupt threshold that can be configured by writing to UART_CTL2 [6:0]. In FIFO interrupt mode with flow control, the programmer must also ensure that the Receive_Hardware_Flow_Control_Threshold (UART_CTL1 [6:0]) value is greater than or equal to the Receive_Int_Threshold (UART_CTL2 [6:0]). Otherwise, FIFO operation stalls.

In FIFO interrupt mode, the processor is informed of the status of the receiver and transmitter by an interrupt signal. The rx_fifo_full interrupt is set when the number of RX FIFO data Byte is larger than the RX interrupt full threshold (threshold value). It is auto cleared when the condition disappears. The tx_fifo_empty interrupt is set when the number of TX FIFO data Byte is less than the TX interrupt empty threshold (threshold value). It is auto cleared when the condition disappears. The two interrupt sources can be masked by writing “0” to UART_IEN register bit 0 and bit 1. The interrupt signal instructs the local hosts to write data to TX FIFO or read data from RX FIFO.

Note that in the cases of the UART flow control being enabled along with the interrupt capabilities, the user must ensure that the UART flow control FIFO threshold is greater than or equal to the receive FIFO threshold.

The TX and RX FIFO may be accessed by DMA. Writing to control register UART_CTL1 bit 15 will enable DMA mode. In receive mode, a DMA request is generated as soon as the receive FIFO reached its threshold level defined in UART_CTL2 [6:0]. This request is deserted when the number of Bytes is less than the threshold level. In transmit mode, a DMA request is automatically asserted when the transmit FIFO is almost empty.

In this mode, when RX FIFO is almost full (the number of RX FIFO data Byte is larger than the RX interrupt threshold value), it will send the receive request to DMA controller, and when TX FIFO is almost empty, it will send the transmit request to DMA controller.

8.11.3.3 Flow Control

The UART module support hardware and software flow control. The hardware flow control should use rtsn/ctsn signal to indicate the flow status. The software flow control use command which integrated in data stream in the txd/rxd signal to indicate the flow control status. The hardware flow control has two mode:

- Auto flow control.
- Software configured hardware flow control.

When the UART_SFCTL register bit1 is set to 1, the software flow control enabled, or if the UART_SFCTL[1] is 0 and if the UART_CTL1[8:7] is set to 2'b11 the auto hardware flow is enabled, or if the UART_CTL[8:7] is 2'b00, the rtsn is set by software, and the receive flow control is disabled. But UART can't literally switch flow control mode when UART start to work.

8.11.3.3.1 Hardware Flow Control

The UART module supports both auto hardware flow control and software configured hardware flow control. When UART_CTL1 register bit 7 is set to “0”, software will control the signal uart_rtsn by writing UART_CTL0 register bit 6; if “1”, the signal uart_rtsn will be controlled by hardware itself, called auto-RTS.

Auto hardware flow control is composed of auto-CTS and auto-RTS. Auto-CTS and auto-RTS can be enabled/disabled independently by programming UART_CTL1 [8:7].

Auto-RTS data flow control originates in the receiver block. The receiver FIFO trigger levels used in auto-RTS are stored in the UART_CTL1 [6:0]. RTS is active if the RX FIFO level is below the HALT trigger level. When the receiver FIFO HALT trigger level is reached, uart_rtsn is deserted. The sending device (external UART device) may send an additional Byte after the trigger level is reached because it may not recognize the desertion of

RTSn until it has begun sending the additional Byte. The assertion requests the sending device to resume transmission. In this case, `uart_rtsn` is an active-low signal.

With auto-CTS, `uart_ctsn` must be active before the module can transmit data. The transmitter circuitry checks `uart_ctsn` before sending the next data Byte. When `uart_ctsn` is active, the transmitter sends the next Byte. The auto-CTS function reduces interrupts to the host system. When auto-CTS flow control is enabled, the CTS state changes need not trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result. In this case, `uart_ctsn` is an active-low signal.

8.11.3.3.2 Software Flow Control without CTSn and RTSn

The UART module support software flow control without use CTSn and RTSn signals. When the `UART_SFCTL` register **bit[1]** is set to 1, the software flow control is enabled. The user also can config the `UART_SFCTL` register **bit[0]** to select the mode of the software flow control function.

The process of the software flow control is showing as the [Figure 8-12](#). The receiver endpoint use the receive FIFO index to judge whether to suspend the transmission. When the receive FIFO index is larger than the upper threshold `CTRL1[6:0]` (the same as hardware flow control threshold), the receive endpoint will send XOFF command by `uart_tx` to the transmit endpoint, then the transmit will be suspend. When the receive FIFO index is smaller than the lower threshold `DSPWAIT[16:10]`, the receive endpoint will send the XON command to transmitter to restart the transmission. When configure the register must guarantee the upper threshold `CTRL1[6:0]` bigger or equivalent the lower threshold `DSPWAIT[16:10]`. Vice verse, when transmitting data, the `uart_rx` will receive the XOFF and XON command to suspend and restart the transmission. So the UART module will realize the control flow without use CTSn and RTSn signals, it can reduce the usage of IO resource.

Since the software flow control will send command to represent the flow control status, there will be several data have the same value as the XCMD(XON/XOFF), in order to prevent transmission error, the data or the command should be escaped before transmitting or after the receiving. So it will use another command ESC to represent the data status.

The UART module supports two mode for escaping. The first mode is send XCMD as the normal data, and use ESC command and the inverted data to represent the transfer data that has the same value of the XCMD and ESC. The second kind is send the data that has the same value of the XCMD as normal uart transmitting, and send XCMD with a ESC prefix. The format example is shown as follow.

Suppose using 0x11 to represent XON, 0x13 to represent XOFF, 0x77 to represent ESC:

- When use mode 0 to transmit data 0x11 0x13 0x77, the data should be escape to 0x77 0xEE 0x77 0xEC 0x77 0x88. When transmit XON, it should be transmit 0x11.
- When use mode 1 to transmit data 0x11 0x13, the data can be transmitted without escape, when transmit XON, the transmit data should be 0x77 0x11, when transmit data 0x77, the data should be escaped to 0x77 0x77.

Figure 8-12 Software Flow Control Processing Flow

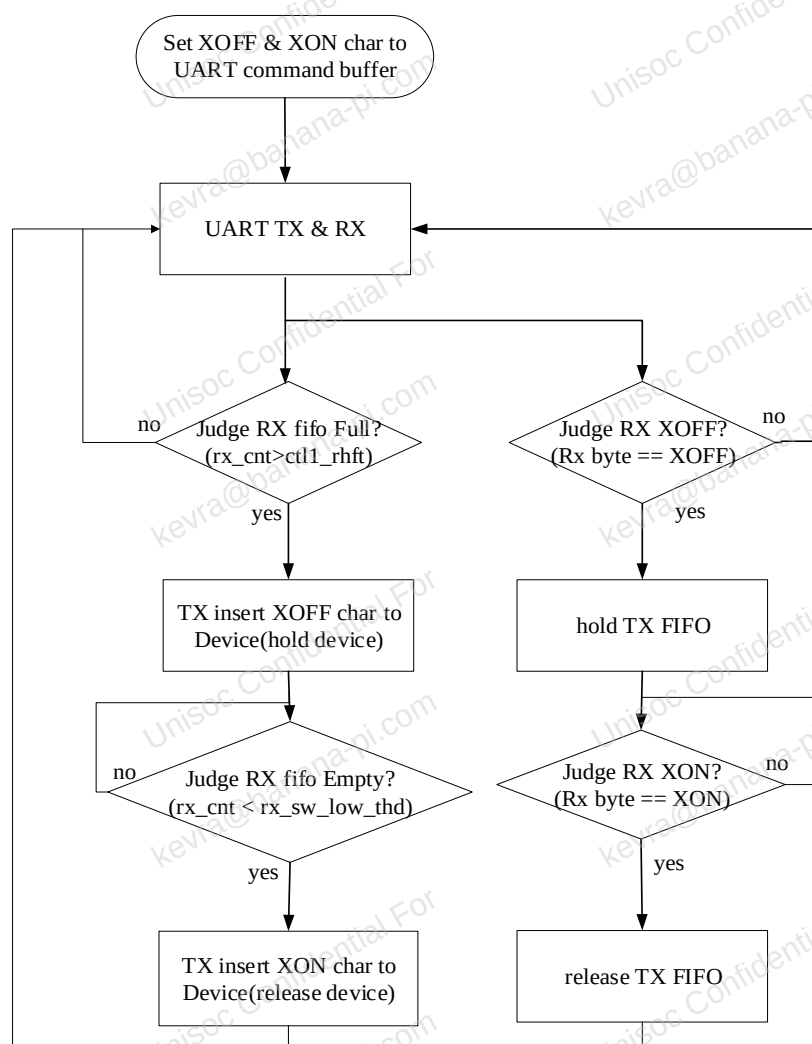


Figure 8-13 Software Flow Control Data Stream (Mode0/Mode1)

Mode 0	DAT0	XCMD	DAT1	DAT2	ESCP	~DAT3	DAT4	XCMD	DAT5	
Mode 1	DAT0	ESCP	XCMD	DAT1	DAT2	DAT3	DAT4	ESCP	XCMD	DAT5

8.11.3.3 Flow control diagram

Figure 8-14 Diagram of tx flow control

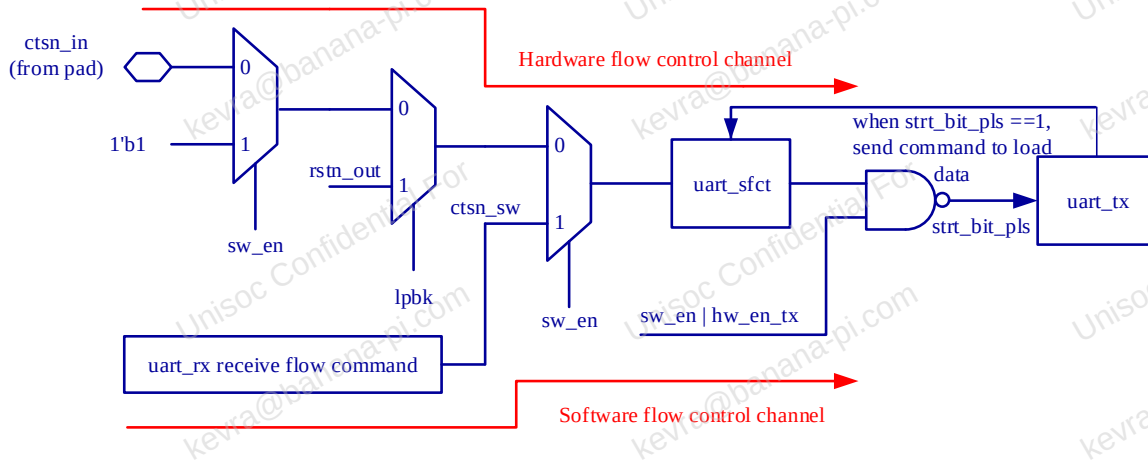


Figure 8-14 is the architecture diagram of TX flow control, as it depicted in the graphic, TX flow control contains two channels, hardware flow control and software flow control.

- In hardware flow control, the control signal ctsn come from other remote RX rstn, when ctsn pull high and the software flow control disable, hardware flow control of TX enable the ctsn signal will propagate to uart_sfct module. When uart_sfct mode sampling the ctsn high will stop to fetch data from TX FIFO to send uart_TX module.
- In software flow control, when locate rx receive the flow control command (XON, XOFF), then translate the command to generate the flow control signal ctsn_sw, and the software flow control enable ctsn_sw will send to uart_sfct module, When uart_sfct mode sampling the ctsn high will stop to fetch data from tx FIFO to send uart_tx module.

According to the diagram if the software and hardware flow control both enable, the flow control will select the software flow control channel, the software flow control has the high priority.

Figure 8-15 Diagram of rx flow control

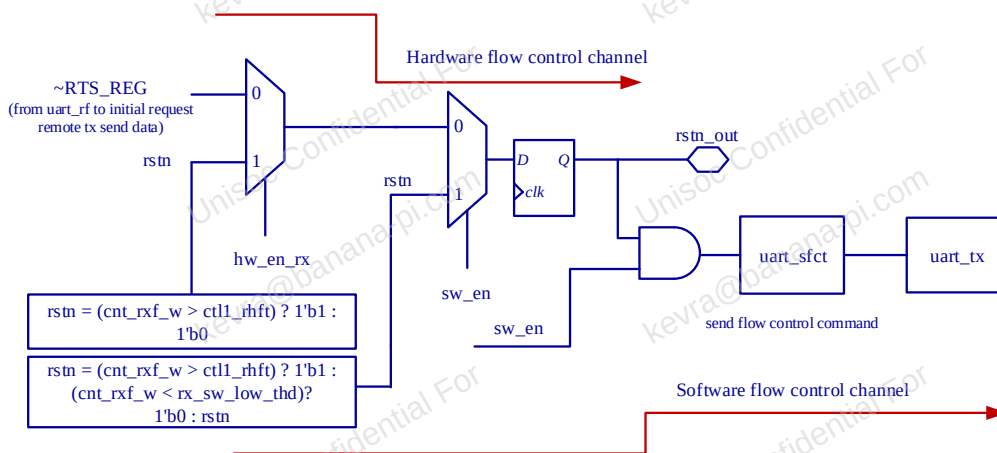


Figure 8-15 is the architecture diagram of RX flow control, as it depicted in the graphic. As the same with the TX flow control, the flow control constrains hardware and software flow control channels. Initial configure the uart control register the RTS_REG should pull down, to initial request remote TX send data, and if hardware of rx and software flow control both disable the flow control can be controlled by cup.

- In hardware flow control, when the RX FIFO received the data beyond the receive threshold, the rstn will pull high, and hardware flow control enable software flow control disable, the rstn will send to the pad out directly to the other remote tx to stop to send data.
- In software flow control, when the RX FIFO received the data beyond the receive threshold, the rstn will pull high, and software flow control enable, the rstn will send to uart_sfct module. The uart_sfct module sampled the rstn was high will stop to prefetch the data from tx_fifo, and send the stop command (XOFF) to uart_tx module to send out to stop the other remote tx, otherwise will send start command (XON) to uart_tx module.

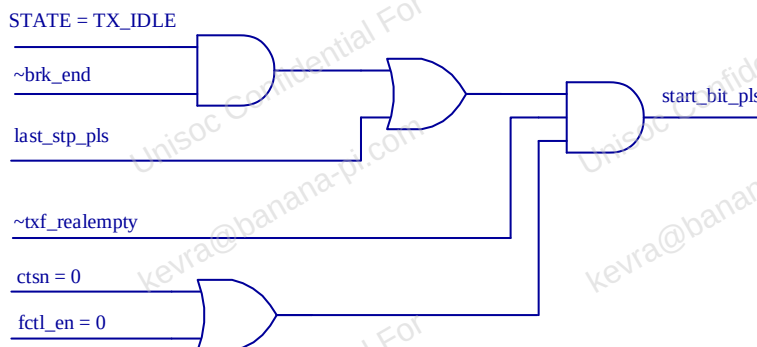
8.11.3.4 RX and TX module

Figure 8-16 depicted the uart send data state diagram and start signal generates circuits. The uart tx module will start to load data under send if the start_bit_pls full high. For initial the state at TX_IDLE and brk_end is “0”, and apb bus write data to the tx FIFO the txf_realempy will pull down, then the start_bits_pls set “1”. When start signal start_bits_pls set “1” the data from tx_fifo will load to the shift register, and the state will change to tx_satart and then under the baudrate control the state to change, until the load Byte data has be send.

When the one Byte data has be send, will generate a last_stp_pls, and then will generate start_bit_pls to load next Byte data to send. If the brk_end CTRL0[7] set to “1” and TX FIFO is real empty and TX is idle, setting this bit forces the TX data output low.

When the flow control signal ctsn is high and flow control is enable will stop the start_bit_pls and stop to send the data.

Figure 8-16 Uart_tx start bit pulse and uart_tx receive control flow



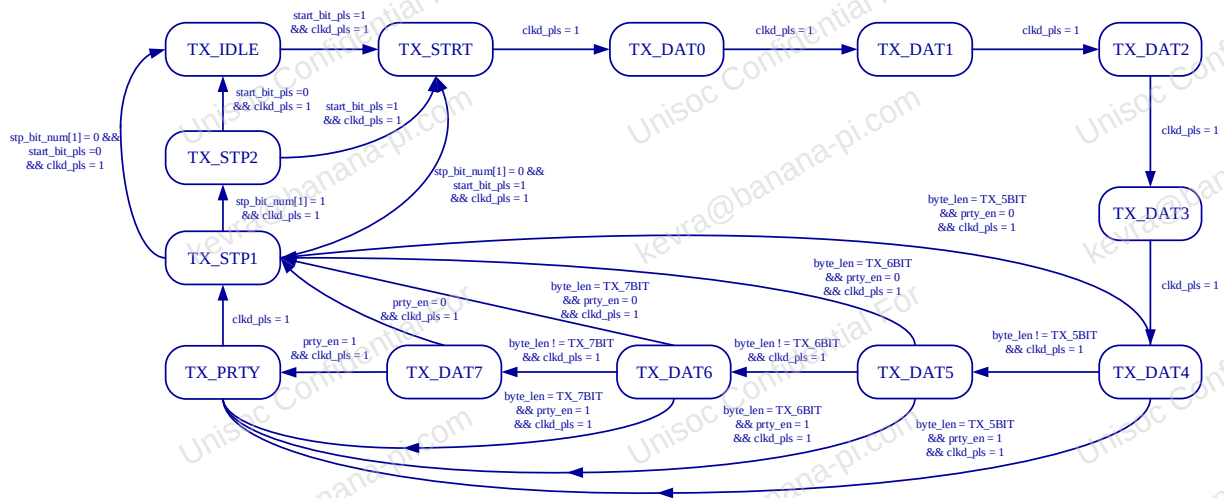
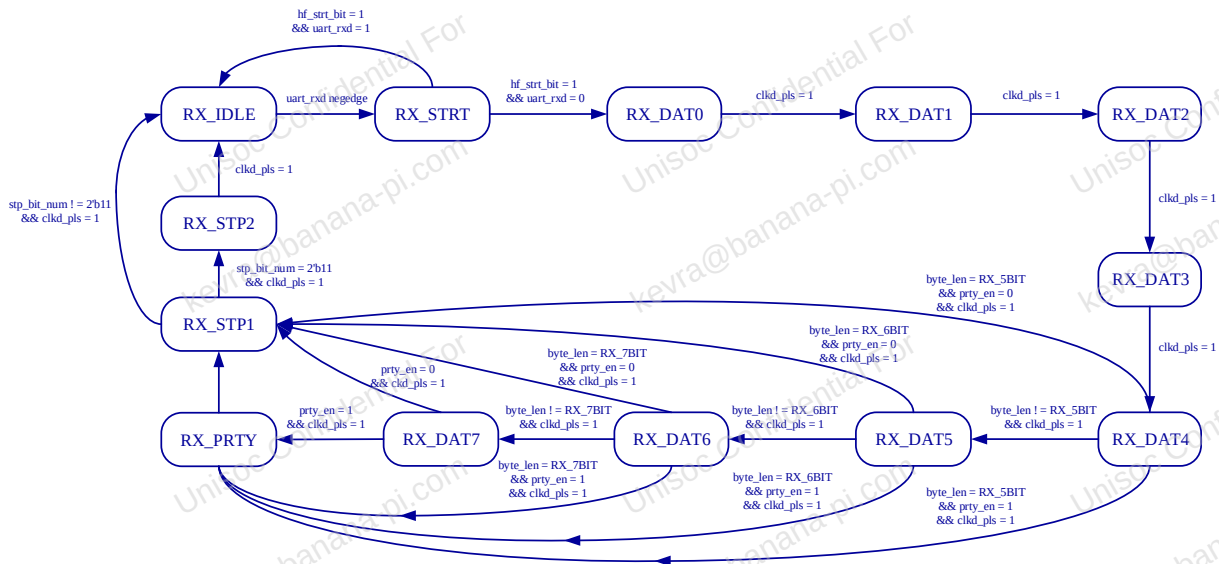


Figure 8-17 depicted the uart rx module receive data state flow diagram, when the `irda_en` negedge come the state will change to `RX_IDLE` state and ready to receive `irda` data, `irda_en` configured by `CTRL0[12:11]`, otherwise when the `rx_d` negedge generated, the state will change to `RX_START`, and the baudrate drives the state change and sampling data from `rx_d`. During sampling the data, if the data exist the frame or parity error the rx will generate the frame or parity error. If the `rx_d` always zero and can't accord with the uart data format, the break detect will generate.

Figure 8-17 Uart_rx flow control



8.11.3.5 Interrupt sources

UART provides a TXF empty interrupt threshold that can be configured by writing UART CTL2 [14:8]. A `tx_fifo_empty` interrupt will be generated when the number of TX FIFO data Byte is less than the TX interrupt threshold value. It is auto cleared when the condition disappears.

UART also provides a RX FIFO full interrupt threshold that can be configured by writing to UART CTL2 [6:0]. An `rx_fifo_full` interrupt will be generated when the number of RX FIFO data Byte is larger than the RX interrupt threshold value. The interrupt will be auto cleared when the condition disappears.

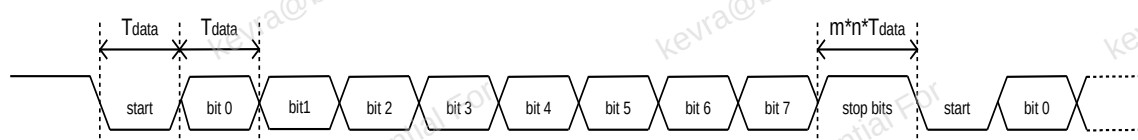
An `rx_tout` interrupt will be generated if the UART has not received data from the input line in a period of time. This period depends on the baudrate and the value of CTL1 [13:9]. If the value of CTL1 [13:9] is '0', the interrupt will never be generated. The interrupt can be cleared by writing "1" to UART_ICLR bit 13.

When having received data, UART generate parity error/framing error if error condition occurs. Also, when having received FIFO overrun, an error interrupt will be generated. Writing to corresponding interrupt clear bit will clear these interrupts.

All interrupt sources in UART module can be disabled and masked by writing to the corresponding registers.

When TX FIFO is empty and TX is idle, setting send break bit forces the TX data output to low.

Figure 8-18 UART timing parameter



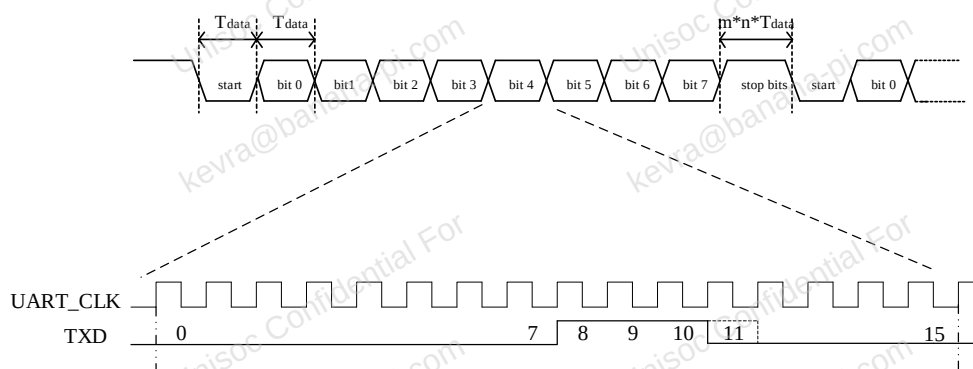
8.11.3.6 IrDA Mode Description

The IrDA can be used to communicate with other IrDA compatible device. IrDA mode is selected by writing "1" to UART CTL0 [15]. The default value of this bit is "0".

IrDA mode is achieved by adding IrDA transmit block and IrDA receive block into UART module. The IrDA transmit block changes the UART transmit serial bits to the IrDA timing, and then sends to TXD port. On the other side, the receive module detects the IrDA timing from the RXD port and changes back to UART bit stream to let the UART receive block to finish the data receiving.

Figure below shows the transmit IrDA timing. The polarity could be changed by UART CTL0 bit 9, and the pulse of a bit may be 3 or 4 high cycle controlled by the UART CTL0 bit 13. In the no polarity invert working mode, when sending "1", the output remains "0". And when sending "0", output follows the IrDA timing as demonstrated in the figure. On the receive data path, the polarity could be change by UART_CTL0 bit 10. The transmission and receive operation could be controlled by the UART CTL0 bit 11 and bit 12. UART_CKD0[15:4] must be set bigger than UART_CKD0[3:0] when using irda mode.

Figure 8-19 IrDA transmit timing



8.11.4 Control Registers

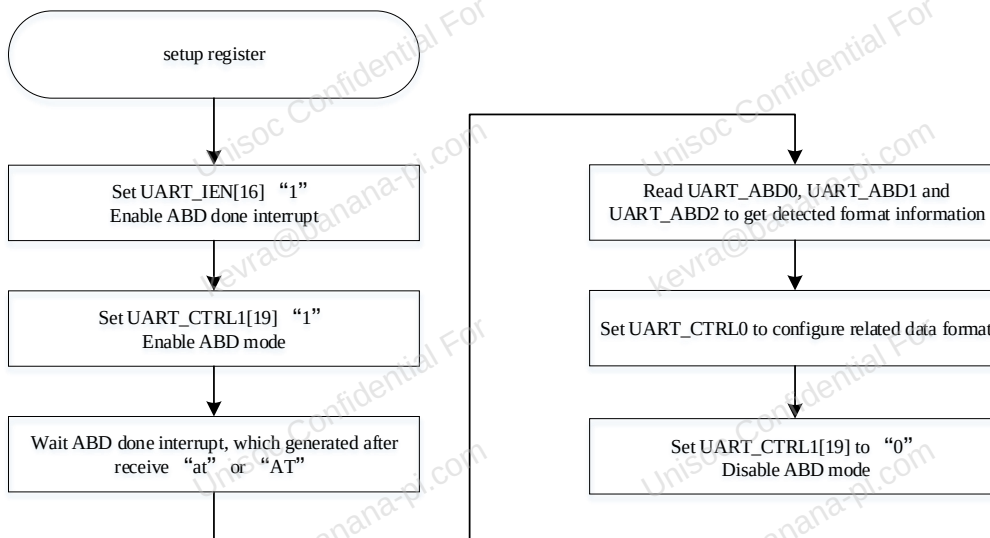
See *UIS7885 AP System Control Registers Instructions*.

8.11.5 Application Notes

Automatic baudrate detection flow

1. The initiator writes to the setup register (global register).
2. The initiator writes UART_IEN register bit 16 to “1” to enable ABD done interrupt.
3. The initiator writes UART_CTRL1 bit 19 to “1” to enable ABD mode.
4. ABD done interrupt will be generated if UART receive the ASCII code of “at” or “AT”. After receiving this interrupt, MCU should clear the interrupt and configure the UART_CTL0 to generate the suitable data format by the information of UART_ABD0, UART_ABD1 and UART_ABD2 registers.
5. Writes UART_CTRL1 bit 19 to “0” to disable ABD mode and enter the UART mode.

Figure 8-20 Uart automatic baudrate detection flow



Working in UART mode with hardware flow control (full duplex)

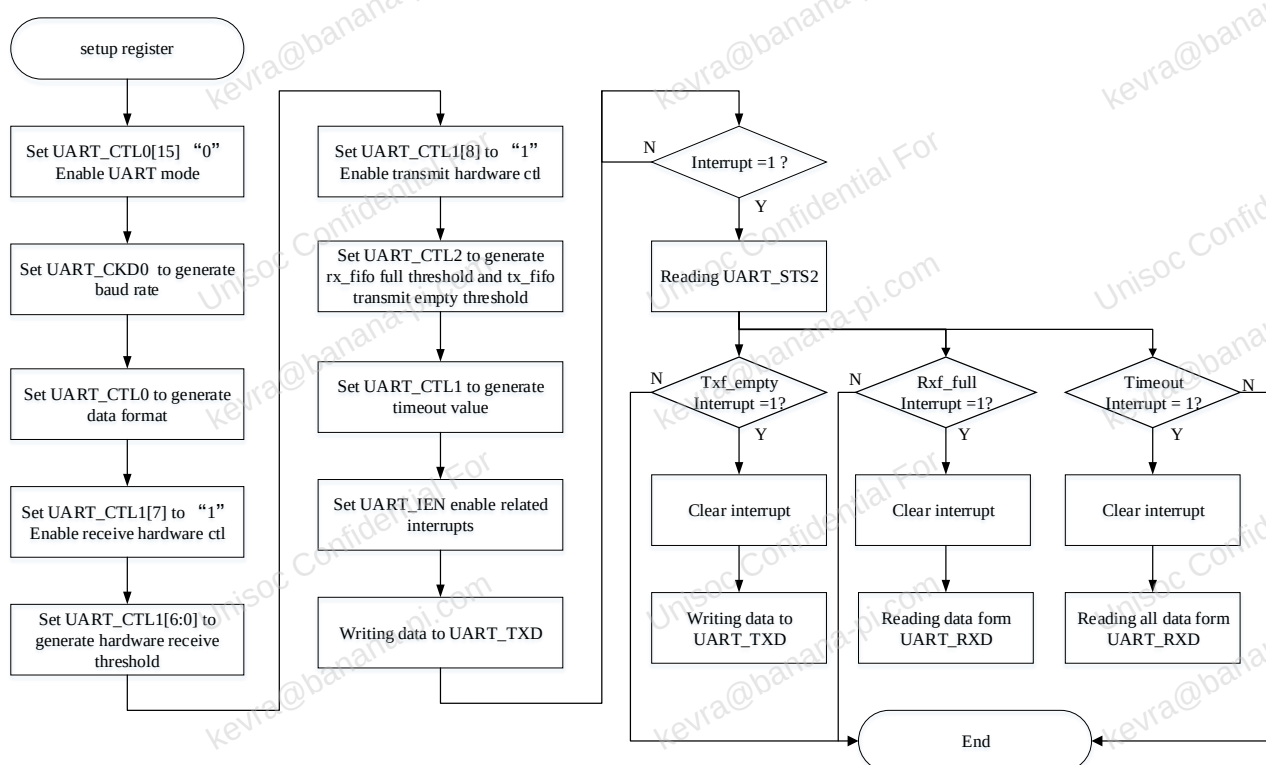
1. The initiator writes to the setup register (global register).
2. The initiator writes the UART_CTL0 register bit 15 to “0” to enable UART mode.
3. The initiator configures UART_CKD0 registers to generate the suitable baudrate. For example, if clk_uart is 26MHz, UART_CKD0 is set to 0xe1, then the UART baudrate is 115kHz.
4. The initiator configures UART_CTL0 to generate the suitable data format, such as parity enable and parity style, Byte length, and stop bit number.
5. Set UART_CTL1 register bit 7 to “1” to enable receive hardware flow control.
6. The initiator configures UART_CTL1 register bit [6:0] to suitable value.
7. Set UART_CTL1 register bit 8 to “1” to enable transmit hardware flow control.
8. The initiator configures UART_CTL2 to generate receive threshold and transmit threshold.
9. The initiator writes UART_CTL1 register to configure receive timeout value.
10. Enable related interrupts by writing UART_IEN register.

11. Writing data to UART_TXD.

Once the TX FIFO is not empty:

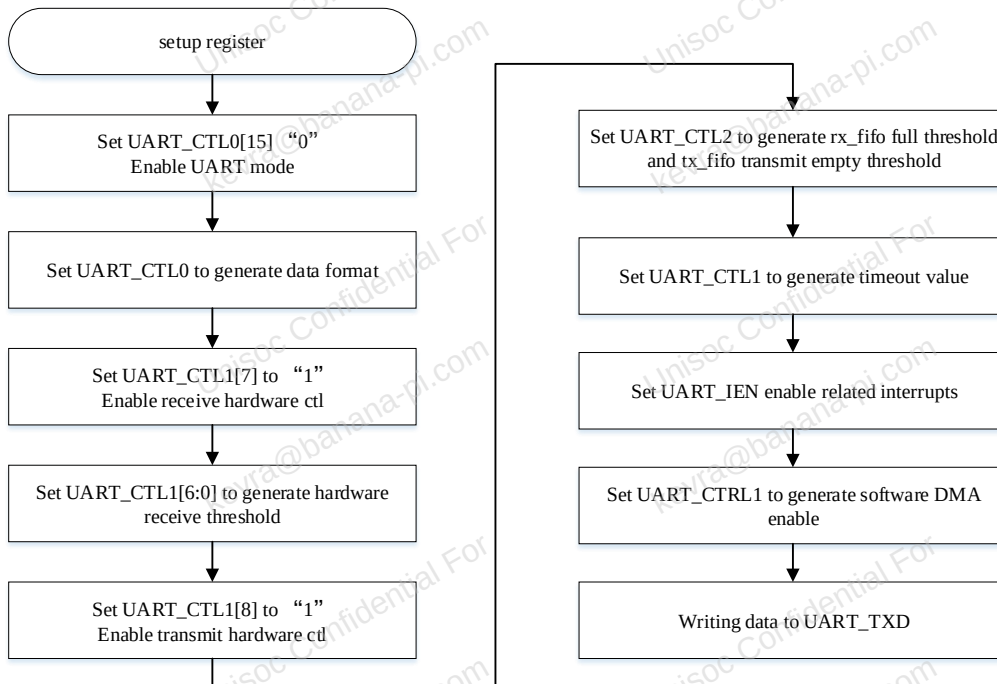
1. Data in TX FIFO will appear on TX line in sequence.
2. If data number in the TX FIFO is less than the data empty threshold value, a txf_empty interrupt will be generated. After receiving this interrupt, MCU should write more data into the TX FIFO. After data number in the TX FIFO is more than the data empty threshold value, MCU should clear the interrupt.
3. If data number in the RX FIFO is more than the data full threshold value. An rxf_full interrupt will be generated. After receiving this interrupt, MCU should read data from RX FIFO. After data number in the RX FIFO is less than the data full threshold value, MCU should clear the interrupt.
4. If receiving RX_TOUT interrupt, read all data in the RX FIFO.

Figure 8-21 Uart mode with hardware flow control



Working in UART mode with hardware flow control (full duplex) with DMA mode

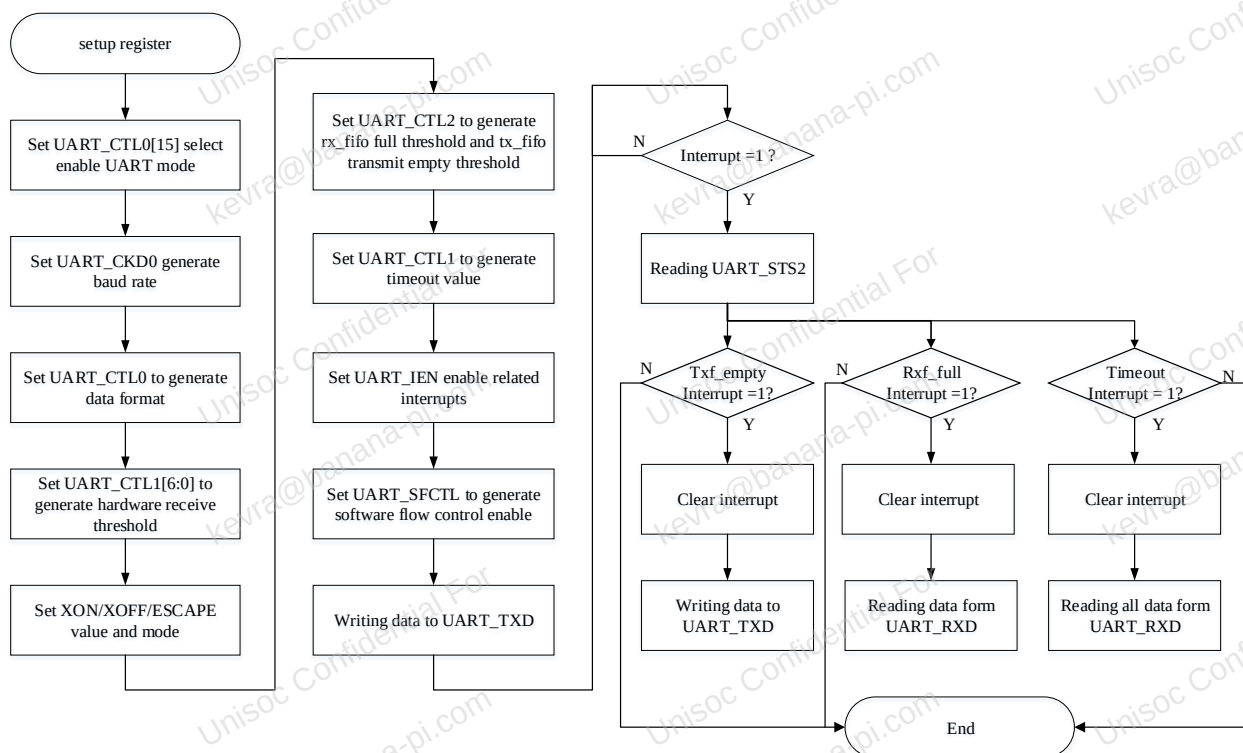
Figure 8-22 Uart with DMA mode with hardware flow control



When use the software flow control without ctsn and rtsn signal (full duplex)

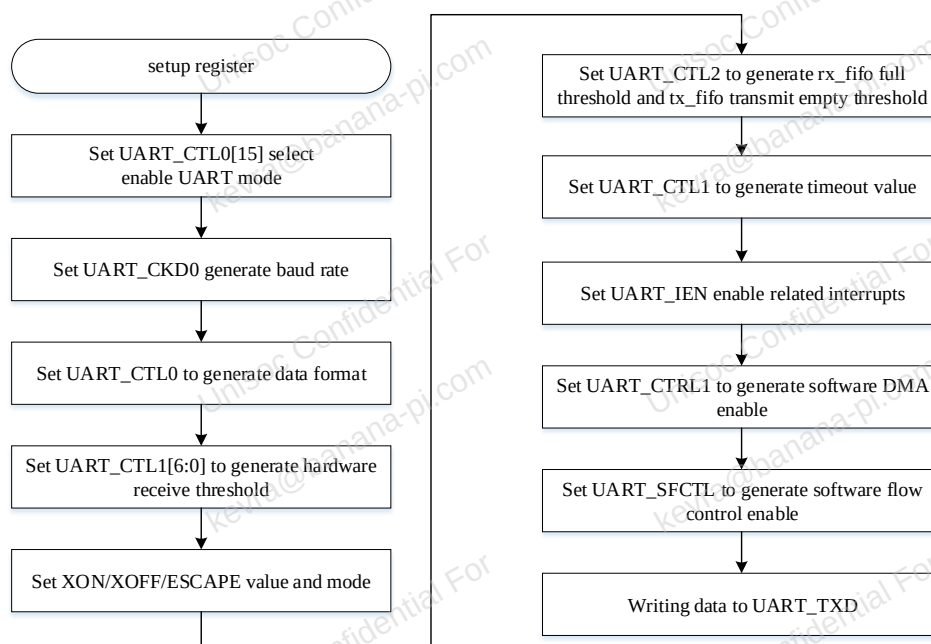
1. The initiator writes to the setup register (global register).
2. The initiator writes the UART_CTL0 register bit 15 to "0" to enable UART mode.
3. The initiator configures UART_CKD0 registers to generate the suitable baudrate. For example, if clk_uart is 26MHz, UART_CKD0 is set to 0xe1, then the UART baudrate is 115kHz.
4. The initiator configures UART_CTL0 to generate the suitable data format, such as parity enable and parity style, Byte length, and stop bit number.
5. The initiator configures UART_CTL1 to generate the Recieve_Hardware_Flow Control_Threshold (software flow control also used this value to determine the rx_fifo is full).
6. The XON/XOFF/ESCAPE register should be configured before the enable of software flow control and mode selection.
7. The initiator configures UART_CTL2 to generate the TX FIFO data empty threshold and RX FIFO data full threshold (must guarantee the value is lower than or equal to Recieve_Hardware_Flow Control_Threshold).
8. The initiator writes UART_CTL1 register to configure receive timeout value.
9. The initiator configures UART_IEN to make TX FIFO empty and RX FIFO full interrupt enable.
10. The software flow control enable bit should be configured before starting a software flow control transition.
11. Writing data to UART_TXD.
12. When UART_IRQ set, reading data from UART_STS2 to configure the related interrupt.

Figure 8-23 Uart mode with software flow control



When use the software flow control without ctsn and rtsn signal (full duplex) with DMA mode

Figure 8-24 Uart with DMA mode with software flow control



8.12 SPI_HS

8.12.1 Overview

SPI_HS is a serial synchronous communication interface. SPI block enables the device to interface with SPI peripherals (i.e. A/D converters, display drivers, EEPROMs etc.) It consists of a serial shift register with serial data input, serial data output and serial shift clock. The shift clock can be selected from either an interval source or an external source. Operating the SPI with the internal clock source is called the Master mode of operation. Similarly, operating the SPI with an external shift clock is called the Slave mode of operation.

SPI module support MICROWIRE/PLUS mode, sync mode, s8 mode and 3-wire mode. Only in MICROWIRE/PLUS mode, master and slave both be supported. This document defines function and configuration of SPI block. SPI_HS include two blocks, SPI controller and SPI phy.

8.12.2 Features

- Compliant with the SPI standard
- Support MICRO/PLUS mode, SYNC mode, S8 mode and 3-wire mode
- Serial clock with programmable frequency , polarity
- Wide selection of SPI word lengths ranging from 1 to 32 bits
- Two 32-word-deep FIFOs minimizes processor overhead at high data rates
- Support master / slave mode
- Data transmit or receive with MSB first or LSB first
- Transmit only/ receive only / transmit and receive mode
- For only receive mode , the length of words ready to receive form the slave can be programmed
- The intervals of two SPI frames can be programmed
- Support DMA
- Interrupts for FIFO events and transfer events
- The polarity and position of the sync can be configured as slave transmit phase adjustable
- TX/RX FIFO address can be reset
- Support csn input in slave mode
- Little endian default
- Support higher speed data transfer mode for slave
- Support higher speed data transfer mode for master
- Support 3-wire 9bits and 4-wire 8bits SPI mode for LCD interface

8.12.3 Function Description

The SPI master mode supports communication with one independent SPI devices. SPI initiates a data transfer on the data lines (spi_do and spi_di) and generate clock (sck) and control signals(spi_ncsi).

Write data to this Spi_txd register initiates a character transmission through TX FIFO, the pointer txf_wr_address will increase. All data ready to send must be written into the TX FIFO first, then occurs in transmit line.

Data sampled from receive line is placed in RX FIFO first. When the data number in the RX FIFO is more than the receive data full threshold value, an rxf_full interrupt is generated. MCU will read data from RX FIFO as soon as receiving the interrupt.

Figure 8-25 Diagram of SPI Master

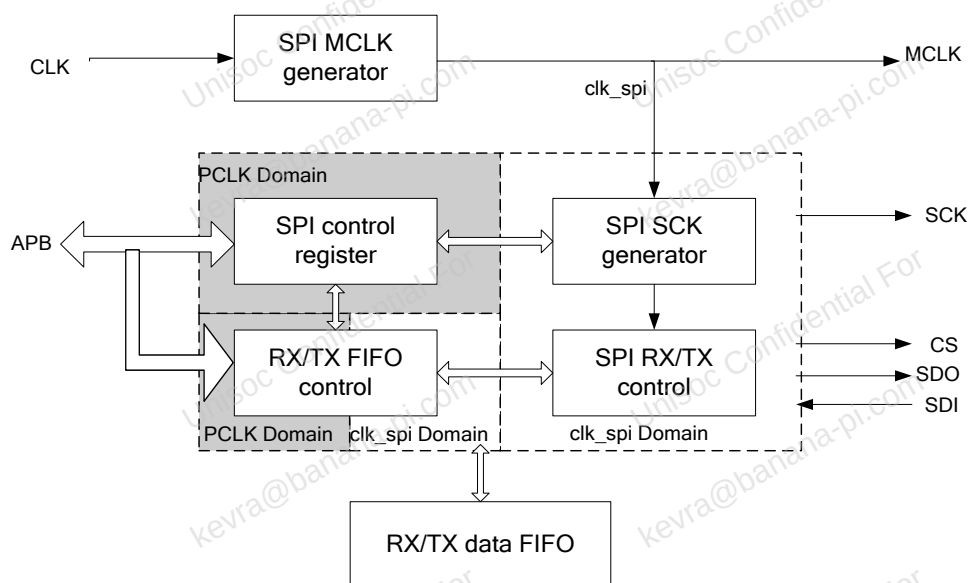
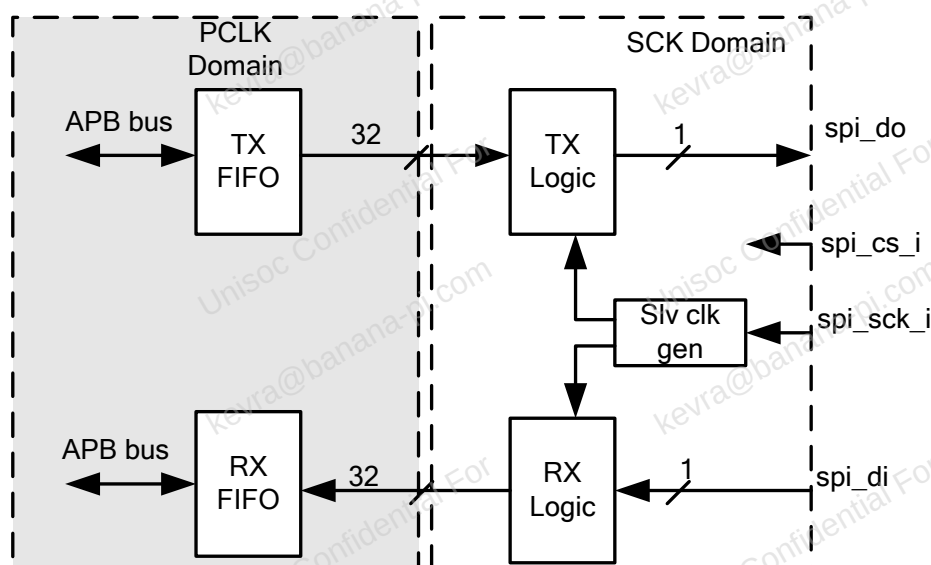


Figure 8-26 Diagram of SPI Slave



8.12.3.1 MICROWIRE/PLUS work mode

MICROWIRE/PLUS is an enhancement of the MICROWIRE™ synchronous serial communications scheme, originally implemented by National Semiconductor's COP400 family microcontrollers. It has three wires, SI (serial input), SO (serial output), and SK (serial clock). The input data on the SI is shifted high order first into the chip; the output data is shifted out high order first from the Most Significant Bit (MSB) on SO. The SK clock is generated internally for the master mode. One burst has 8 data bits. The input data is captured on the rising edge of SK. Following is the timing diagram.

Figure 8-27 MICROWIRE/PLUS Timing (CPOL=0, CPHA=0)

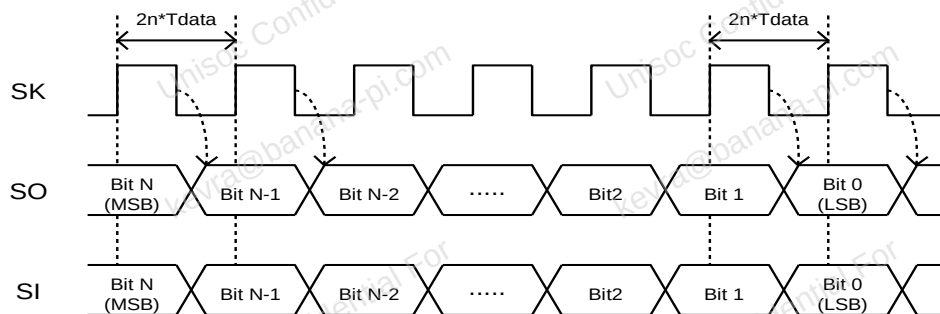


Figure 8-28 Alternated Phase SK Clock Timing (CPOL=0, CPHA=1)

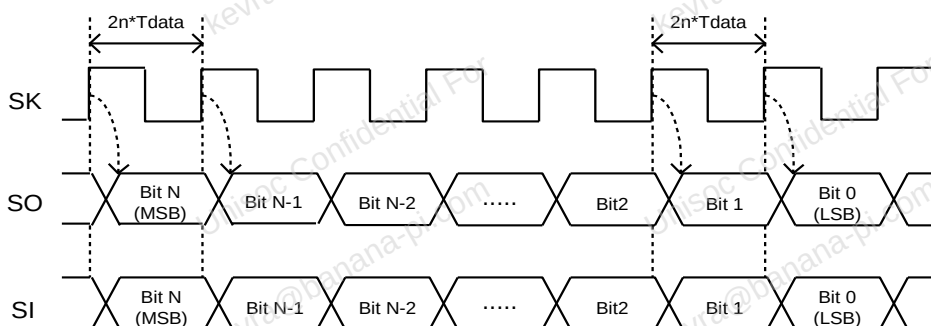


Figure 8-29 MICROWIRE/PLUS Timing (CPOL=1, CPHA=0)

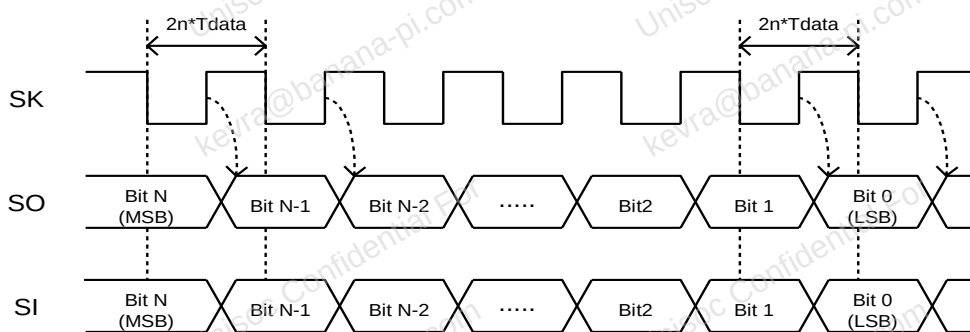
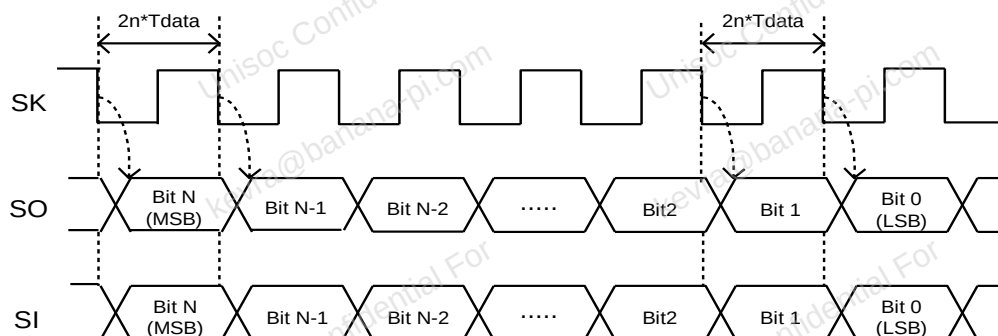


Figure 8-30 MICROWIRE/PLUS Timing (CPOL=1, CPHA=1)



In the normal mode data is shifted in on the rising edge of the SCK clock and data is shifted out on the falling edge of the SCK clock. It is also allow an additional Alternate SCK Phase Operation. In the alternate SCK phase operation, data is shifted in on the falling edge of the SCK clock and data is shifted out on the rising edge of the SCK clock.

To configure SPI for this mode, just enable SPI clock in global control register, program spi_ctl0 control register, enable RX data shift in at SCK rising edge(spi_ctl0[0] = 1), and enable TX data shift out at SCK falling edge(spi_ctl0[1] = 0), set transmit data bit number to 8. Also program spi_clkd according to MCLK and SCK clock rate. Leaving other control register with default value. When put data into TX FIFO, the SPI will send and receive an N bit data at the same time.

The clock phase and TX/RX data timing is programmable to meet other requirements. For example:

- N = 8, CS0, CPOL=0, CPHA = 0 timing as [Figure 8-27](#), set spi_ctl0 = 0x0e22
- N = 8, CS0, CPOL=0, CPHA = 1 timing as [Figure 8-28](#), set spi_ctl0 = 0x0e21
- N = 8, CS0, CPOL=1, CPHA = 0 timing as [Figure 8-29](#), set spi_ctl0 = 0x2e22
- N = 8, CS0, CPOL=1, CPHA = 1 timing as [Figure 8-30](#), set spi_ctl0 = 0x2e21

CPOL=0 means SCK idle phase is low, maps to spi_ctl0[13] = 0; CPOL=1 means SCK idle phase is high, maps to spi_ctl0[13]=1.

CPHA decides SPI working in normal mode or in the alternate SK phase operation. CPHA maps to ctl0_reg[1:0]:

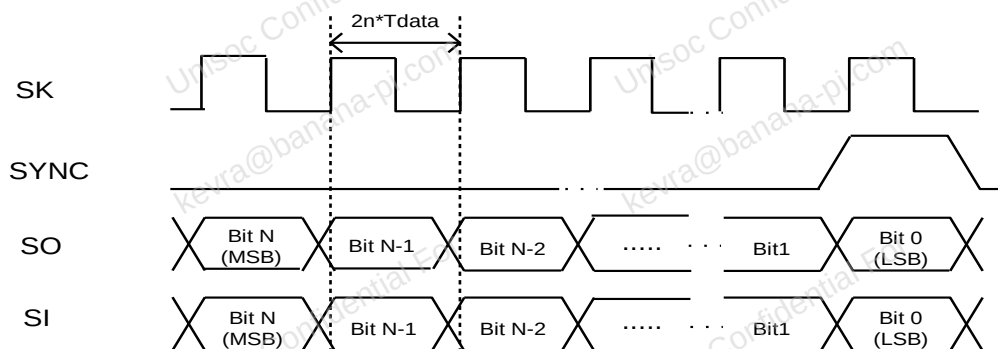
- “10” (CPHA=0) means output data at negedge of clk while receiving data at posedge of clk
- “01”(CPHA=1) means output data at posedge of clk while receiving data at negedge of clk
- “00” and “11” are not available in this mode

8.12.3.2 Synchronous SPI

Besides SCK, SDI, SDO, Synchronous SPI provides the fourth pin as a synchronous signal. It generates single bit width pulse on one of the TX/RX bit position. On slave side, it tells when to latch the received Byte and prepare the next data for transmit.

If enable sync mode by setting spi_ctl0 register bit14, the sync pulse will appear on the last bit. If program spi_ctl1 register bit 4~0 to N, the sync pulse will locates on top of bit N. The SYNC bit can be program output to one of the 4 chip select pins by setting spi_ctl1[11:8].

Figure 8-31 Alternated Phase SK Clock Timing (CPOL=0, CPHA=1)

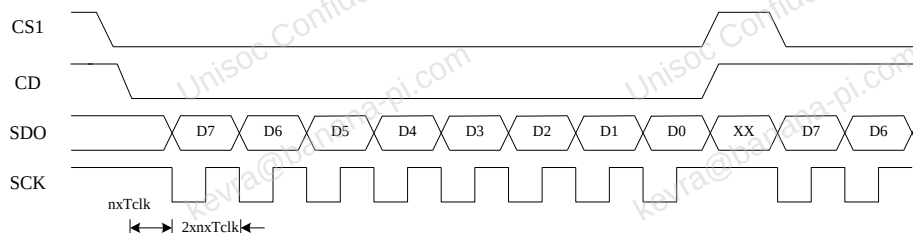


8.12.3.3 4-wire Serial Interface(s8)

The 4-wire serial interface is required by LCD driver like ULTRACHIP UC1607, and only working in write operations mode. Pin CS is used for chip select and bus cycle reset. Pin CD is used to determine the content of the data been transferred. On each write cycle, 8 bits of data, MSB first, are transmitted on falling SCK edges. If $CD=0$, the data Byte will be decoded as command. If $CD=1$, this 8-bit will be treated as data. Pin CD is examined when SCK is pulled low for the LSB(D0) of each burst. The maximum SCK frequency required is 5MHz.

To enable SPI working in this mode, all settings are same as program MICROWIRE except to enable TX data shifting out at SCK rising edge. Also need to set `spi_ctl1` control register bit7, enable s8 mode, and assign the CD signal output from one of the four CS pins by programming bit11~8. Please pay attention to not mapping CD to the same CS pin used for LCD chip select. For example using CS0 as chip select, using cs1 as CD, you should program `spi_ctl0` register bit11~8 "1100" and programming `spi_ctl1` register bit11~8 "0010".

Figure 8-32 4-wire Serial Interface (s8) Timing



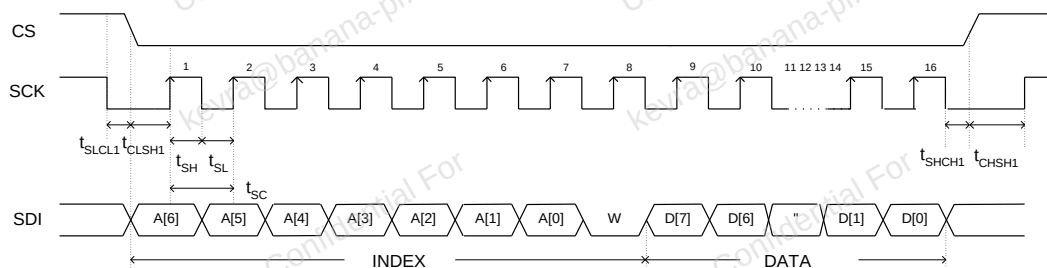
1. SW write 9 bits data to TX FIFO while `ctl0_reg[6:2]` program to 8. Bit 8 is used to control CD signal, and bit[7:0] will be used to transmit to slave.
2. SW Set `SPI_CD_BIT`, this bit will be connected to `SPI_CD` pin, will be used to transmit to slave.

8.12.3.4 3-wire Serial interface

The 3-wire serial interface is required by Sound Generator Chips like OKIML2860. SDO works as an in/out pin because data has to be read/write through the same pin. SCK is the serial clock, output data from SDO change at clock falling edge, input data is sampled at falling edge. See Figure 8-31. There is also an alternate SCK phase operation, data output at clock rising edge and input data is sampled at rising edge. See Figure 8-35. One burst has 16 bits. 7 address bits sent first. Bit8 is command bit. "1" means writing 8 bits data to the slave. "0" means reading 8 bits data from the slave, the read data address in slave is the 7 bits address sent in the same burst.

Figure 8-33 3-wire Serial Interface Timing 1

Data Write Timing 1



Data Read Timing 1

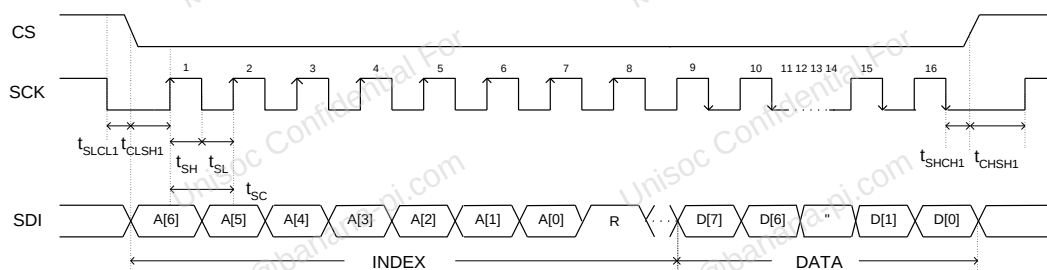
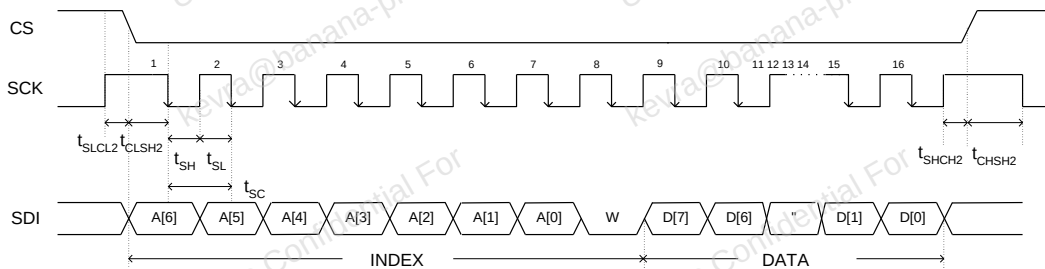
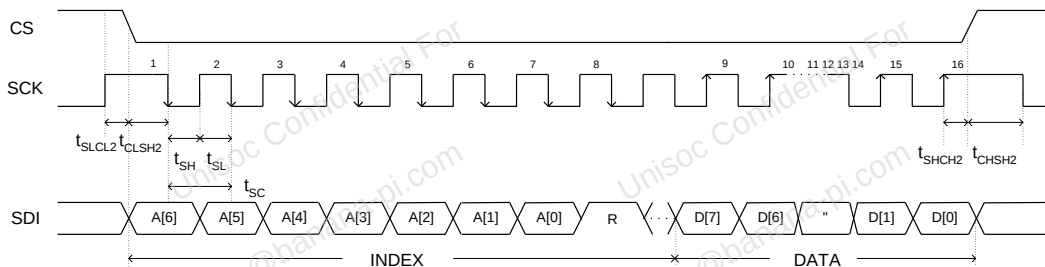


Figure 8-34 3-wire Serial Interface Timing 2

Data Write Timing 2



Data Read Timing 2



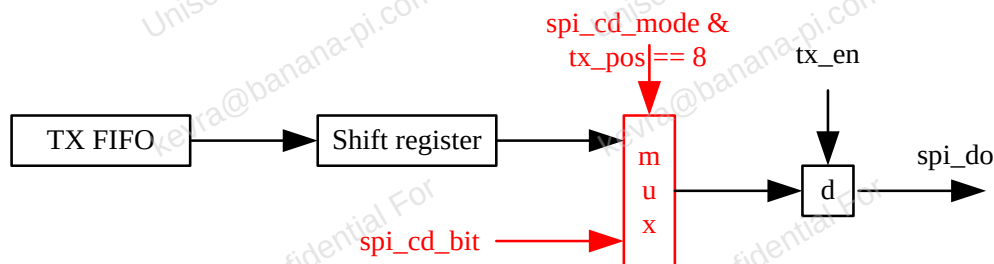
To program SPI in this mode, set `spi_ctl0` control register to 0x0e43, means 16bits sent in one burst, TX at clock falling edge, RX sample at clock falling edge, CS maps to cs0. Set `spi_ctl1` to 0x3028, means to enable 3-wire mode, command bit is bit8. Set `spi_ctl2` to 0x07, the read data starts from bit7. Then put 16 bits data into TX FIFO, and receive 8 bit data from RX FIFO.

If set `spi_ctl1` bit6 to “1”, signal CS will change to low before half cycle of the first edge of SCK, and go high after half cycle of the last edge of SCK .

To program alternate SCK phase timing as [Figure 8-32](#), set `spi_ctl0` bit13 to “1”, other setting are same as timing 1.

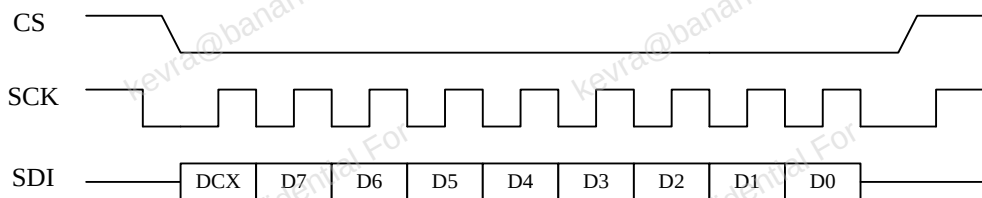
- When set `spi_mode=1 / 2`, cd bit will be send out with data together ,and be put to bit 8 position.

Figure 8-35 spi_cd_mode



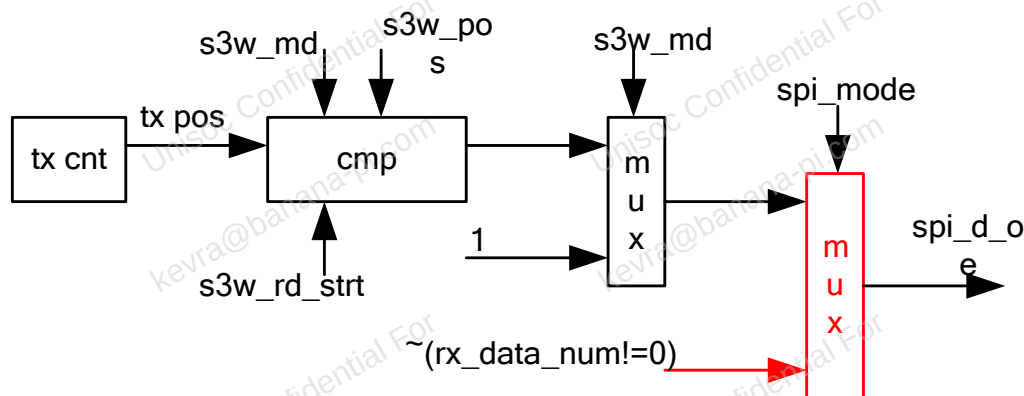
Support cd bit be transmitted with data together. Set SPI_CD_BIT, put it to bit 8 position when transmit out from tx shift register.

Figure 8-36 spi_cd_mode



- When set spi_mode = 1 / 3 , spi_d_oe is controlled by is_rx_only bit, when set is_rx_only bit, spi_d_oe is 0, and spi_d_ie is 1.

Figure 8-37 spi_3w_mode



8.12.3.5 2-data-line Interface

GC9304 SPI supports 4-wire serial interface. And it has 4 wires, which are CS, SCK, SDA and RS.

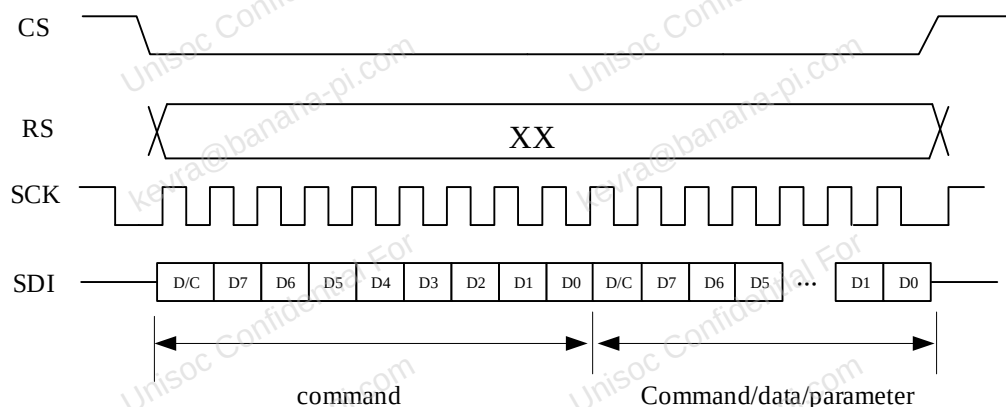
- Write cycle sequence

The write mode of the interfaces means that the host writes commands or data to GC9304C. The 3-lines serial data packet contains a data/command select bit(D/CX) and a transmission Byte. If the D/CX bit is “low”, the transmission Byte is interpreted as a command Byte. Otherwise, the transmission Byte is command register as parameter.

Any instruction in transmission is the MSB is transmitted first.

Host processor drives the CS to low and starts by setting the D/CX bit on SDA. The bit is read by slave on the first rising edge of SCK. On the next falling edge of SCK, the MSB data bit (D7) is set on SDA by the host. On the next falling edge of SCK, the next bit (D6) is set on SDA.

Figure 8-38 2-data-line interfaces of writing data Timing



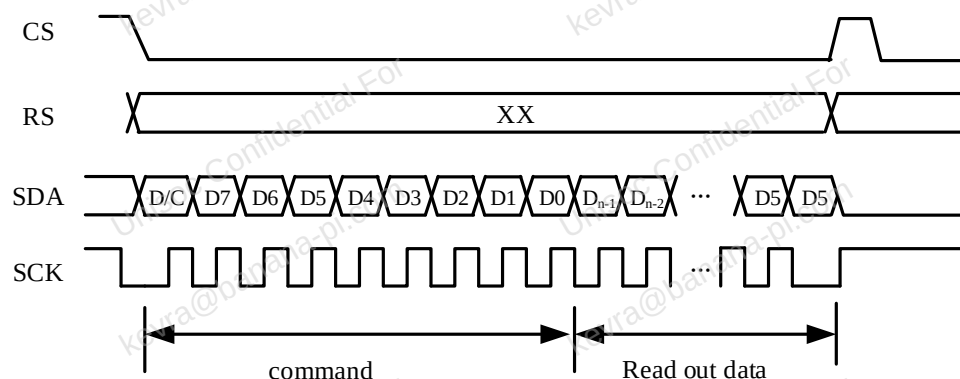
The length of the transaction of a write register/command may be larger than 32 bit, and the maximum length of the transaction of the former design is 32 bit. So the transaction must be divided into 2 parts if the transaction length is larger than 32-bit. The first part is used to write the command, and the second part is used to write the command contents or data/parameter.

- Read cycle sequence

The read register is used in 3-wire mode. The read mode is used to the host reads register's parameter from GC9304C. The host has to send a command (Read ID or register command) and then the following Byte is transmitted in the opposite direction. GC9304C latches the SDA (input data) at the rising edge of SCK. And then shifts SDA (output data) at the falling edges of SCK. After the read status command has been sent, the SDA line must be set to tri-state and no later than at the falling edge of SCK of the last bit. The Read mode has three Bytes of transmitted command data (8/24/32-bit) according to the command code.

When the master read the registers from the slave, it works in the 3-wire mode. The SDA pad is bi-directional. And the RS is no used. But in the former design, the maximum length of a transaction is 32-bit. So the read transaction will be divided into 2 parts which is liked the write transaction. The first part is used to write the command, and second part is used to read the data and parameter. And when the first part is end, the SDA pad will became into a input pad.

Figure 8-39 2-data-line interfaces of reading data Timing



In order to support this case, the design must be added 2 registers to control it. The first register is used to make the hardware work in this mode. And the second register is used to switch the pad direction of the SDA when the command is send in the end. If the transaction is writing, the SDA keep the status as output when

the command is finished (9-bit later). And then the SPI master must change its length register (CHNL_LEN). And the master gets the data from FIFO to send data to the slave.

If the transaction is reading, the SDA pad must be change into input status when the command is finished (9-bit later). And then the SPI master must change its length register (CHNL_LEN) and works in only-receiver mode. And the master gets the data/parameter from the slave.

During the write/read transaction, the CSn must be kept in the low status.

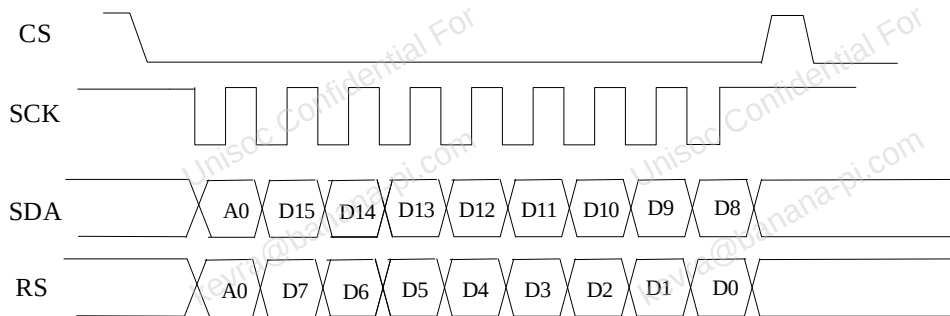
- Write date

Only frame pixel data write transitions are sent in this mode.

The CS enables and disables the serial interface. SCK is the serial data clock. SDA and RS are serial data lines.

Serial data must be output to SDA in the sequence A0, D15 to D8 and RS in the sequence A0, D7 to D0. The master sent the data at the falling edges of SCK. And the GC9304C reads the data at the rising edge of SCK signal. The first bit of the serial date A0 is data/command flag. It must be set to “1”, D15 to D0 bits are display RAM data.

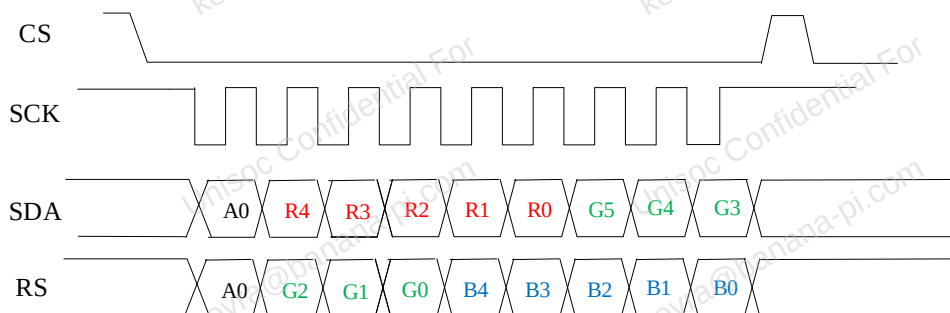
Figure 8-40 2-data-line interfaces of writing pixel data Timing



8.12.3.5.1 RGB565 1pixel/transition

The following Figure is showed that the master transmits a pixel data. It is include 5-bit R, 6-bit G and 5-bit B which follows the A0.

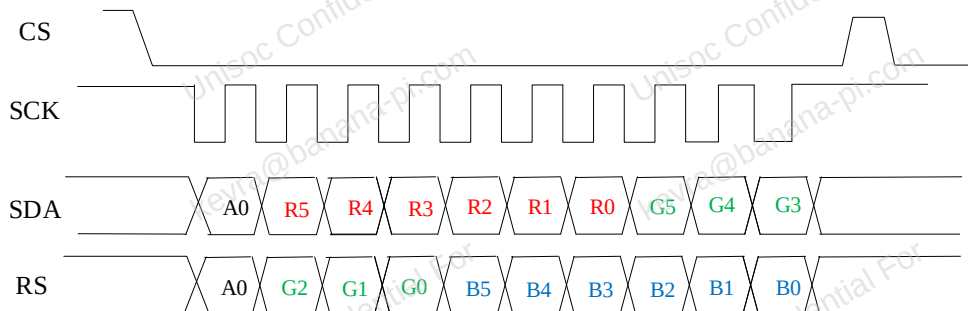
Figure 8-41 RGB565 1 pixel/transition



8.12.3.5.2 RGB666 1pixel/transition

The following Figure is showed that the master transmits a pixel data. It is include 6-bit R, 6-bit G and 6-bit B which follows the A0.

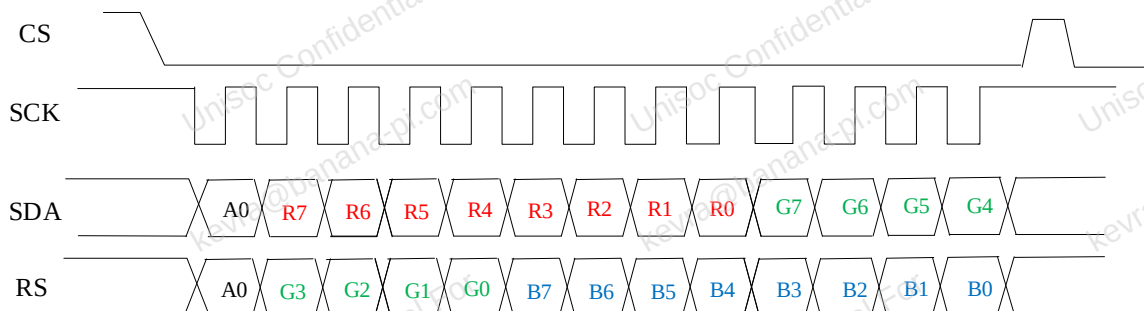
Figure 8-42 RGB666 1 pixel/transition



8.12.3.5.3 RGB888 1pixel/transition

The following Figure is showed that the master transmits a pixel data. It is include 8-bit R, 8-bit G and 8-bit B which follows the A0.

Figure 8-43 RGB888 1 pixel/transition



8.12.3.5.4 RGB data align before the SPI

The data which is written into the SPI TX FIFO is aligned in the below format.

Table 8-5 1 pixel mode

RGB888	[31:24]	[23:16]	[15:8]	[7:0]
XX		R0	G0	B0
XX		R1	G1	B1

RGB666	[31:24]	[23:22]	[21:16]	[15:14]	[13:8]	[7:6]	[5:0]
XX		--	R0	--	G0	--	B0
XX		--	R1	--	G1	--	B1

RGB565	[31:27]	[26:21]	[20:16]	[15:11]	[10:5]	[4:0]
	R0	G0	B0	R1	G1	B1
	R2	G2	B2	R3	G3	B3

Table 8-6 2/3 pixel mode

RGB888	[31:24]	[23:16]	[15:8]	[7:0]
	R1	B0	G0	R0
	G2	R2	B1	G1
	B3	G3	R3	B2

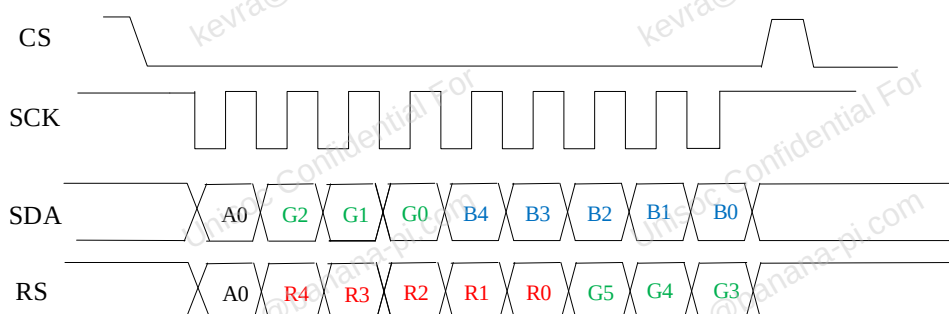
RGB666	[31:30]	[29:24]	[23:22]	[21:16]	[15:14]	[13:8]	[7:6]	[5:0]
	--	R1	--	B0	--	G0	--	R0
	--	G2	--	R2	--	B1	--	G1
	--	B3	--	G3	--	R3	--	B2

8.12.3.5.5 2-data-line switch

For more flexible data format, support 2-data-line switch when set DATA_LINE2_SW (SPI_CTL8[11]) to 1. 2-data-line switch only valid after set DATA_LINE2_EN(SPI_CTL7[15]) to 1.

The following Figure is showed that the master transmits a RGB565 pixel data after set DATA_LINE2_SW, data transfer by SDA and RS has switched. where SDA stands for spi_do and RS stands for spi_di.

Figure 8-44 RGB565 1 pixel/transition(set DATA_LINE2_SW to 1)



8.12.3.5.6 The design changed for 2-data-line mode in write data

When the SPI write data to the slave, there are 2 line data to transfer the data. And the maximum length is in RGB888 mode. The length is 26. And the TX FIFO is 32-bit, so you can fill the TX FIFO into two parts, the lower parts is used to store the SDA data, and the higher parts is used to store the RS data. When the SPI read the data from TX FIFO, and shift the read data to data lines in 2-bit once time.

The pad spi_di is the output at the same time, so you should add the spi_di_oe and spi_di_ie to control the pad (spi_di_o). And the A0 is added by HW automatically, so the spi mode should like the 3-wire 9-bit mode. The following code is showed the logic of spi_di_p.

1. Add a new signal spi_di_p for spi_di pad output
2. Add a new signal spi_di_oe for control spi_di pad output enable
3. Add a new signal spi_di_ie for control spi_di pad input enable

So the write data to the TX FIFO must be aligned again to following the formats which are showed from [Table 8-7](#) to [Table 8-12](#). In RGB565 mode, the APB bus will write 32-bit once time, it means there are 2 pixels are written into the FIFO. And the FIFO will shift the pixel data just in one pixel once time. So in RGB565 mode, the write FIFO should be divided into 2 parts. The first time is used to write the higher bits and the second time is used to write the lower bit.

- RGB888

In this mode, the APB bus write 32bit into the FIFO and write data's sequence is showed as the below table.

Table 8-7 RGB888 mode APB wdata format

No.	[31:24]	[23:16]	[15:8]	[7:0]
APB wdata	XX	R	G	B

The APB bus write data should be aligned as the following format.

Table 8-8 RGB888 mode TXFIFO wdata format

No.	[31:28]	[27:24]	[23:16]	[15:12]	[11:4]	[3:0]
TXFIFO wdata	XX	[11:8]	[7:0]	XX	[23:16]	[15:12]

- RGB666

In this mode, the APB bus write 32bit into the FIFO and write data's sequence is showed as the below table.

Table 8-9 RGB666 mode APB wdata format

No.	[31:24]	[23:22]	[21:16]	[15:14]	[13:8]	[7:6]	[5:0]
APB wdata	XX	--	R0	--	G0	--	B0

The APB bus write data should be aligned as the following format.

Table 8-10 RGB666 mode TXFIFO wdata format

No.	[31:25]	[24:22]	[21:16]	[15:9]	[8:3]	[2:0]
TXFIFO wdata	XX	[10:8]	[5:0]	XX	[21:16]	[13:11]

- RGB565

In this mode, the APB bus write 32bit into the FIFO and write data's sequence is showed as the below table.

Table 8-11 RGB565 mode APB wdata format

No.	[31:27]	[26:21]	[20:16]	[15:11]	[10:5]	[4:0]
APB wdata	R0	G0	B0	R1	G1	B1

The APB bus write data should be aligned as the following format.

Table 8-12 RGB565 mode TXFIFO wdata format

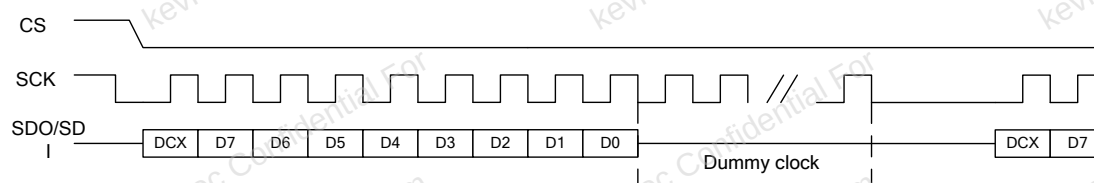
No.	[31:24]	[23:21]	[20:16]	[15:8]	[7:3]	[2:0]
TXFIFO wdata0	XX	[23:21]	[20:16]	XX	[31:27]	[26:24]
TXFIFO wdata1	XX	[7:5]	[4:0]	XX	[15:11]	[10:8]

In this mode, if the APB writes one time into SPI, and the SPI TX FIFO must be written two times.

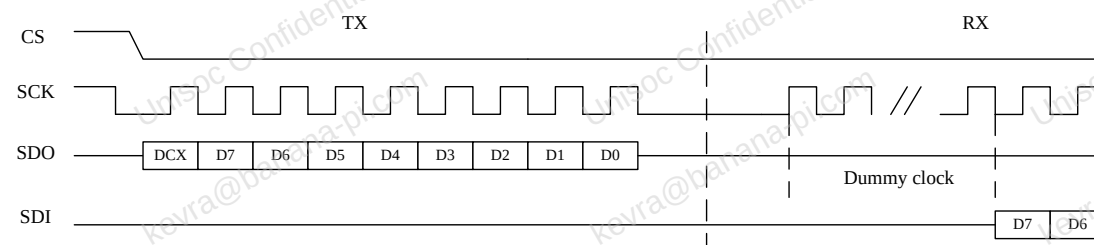
8.12.3.6 Dummy control

SPI controller supports write and read configurable dummy clocks.

- For TX mode, dummy clock number is controlled by SPI_TX_DUMMY_LEN, which set dummy clock number when write LCD.

Figure 8-45 TX Dummy waveform


- For Rx mode, dummy clock number is controlled by SPI_RX_DUMMY_LEN, which set dummy clock number when read LCD.

Figure 8-46 RX Dummy waveform


8.12.3.7 SPI Slave Design

8.12.3.7.1 SPI Slave Clock

Figure 8-47 SPI Slave Clock

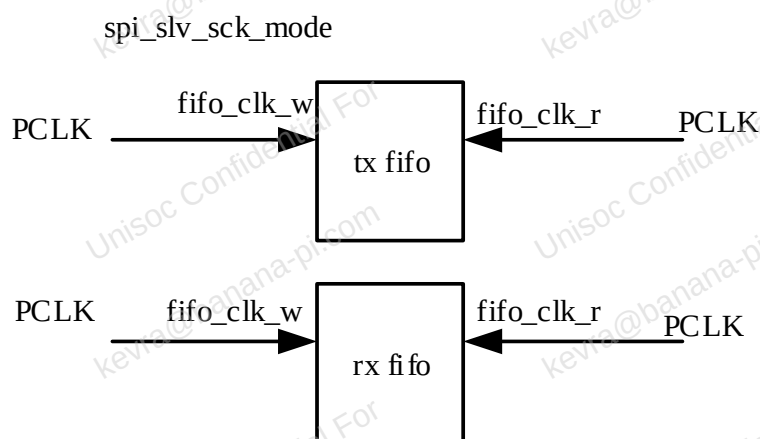
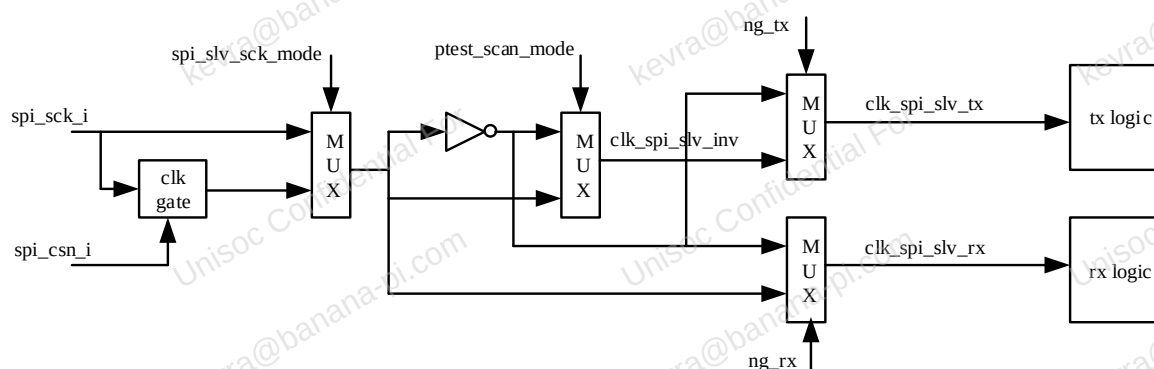
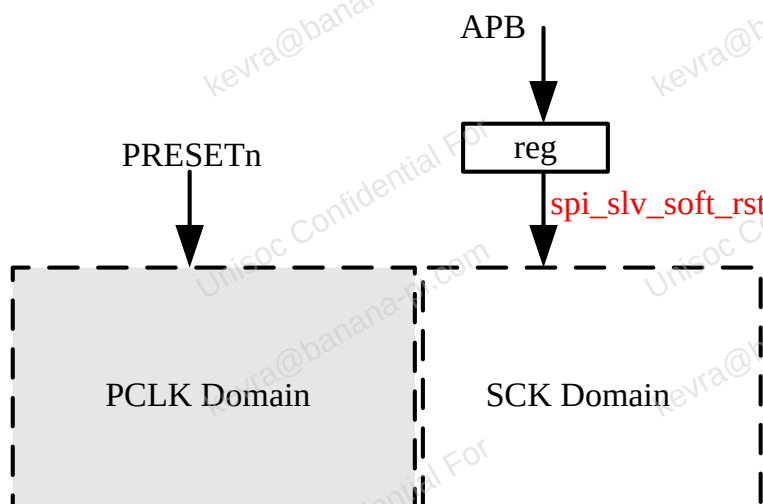


Figure 8-48 SPI Slave TX and RX Clock



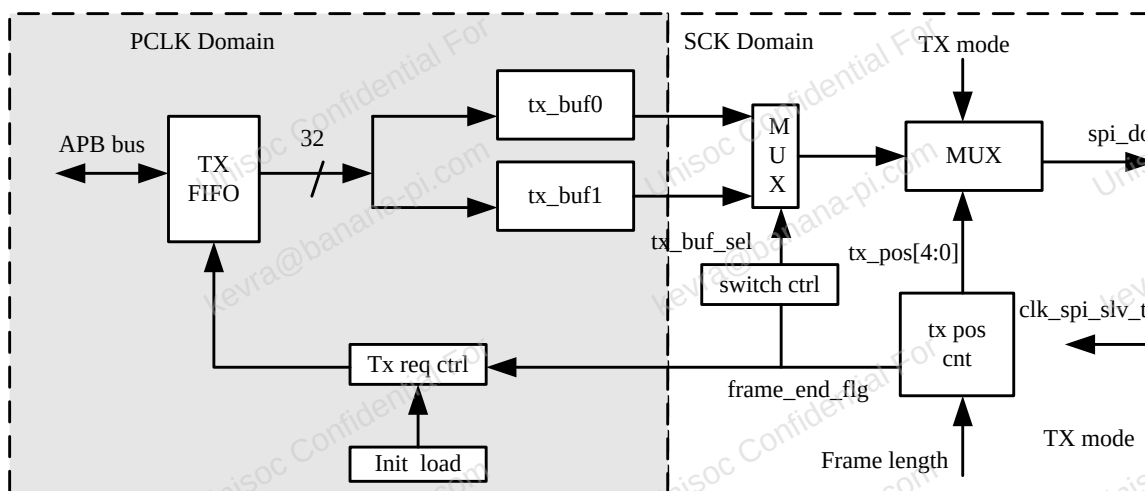
8.12.3.7.2 SPI Slave Reset

Figure 8-49 SPI Slave Reset



8.12.3.7.3 SPI Slave TX

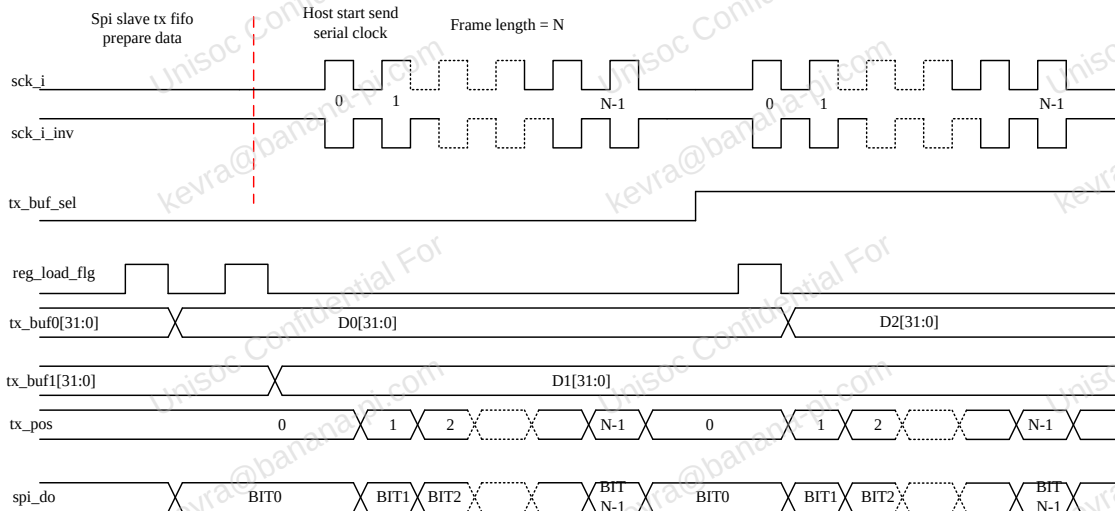
Figure 8-50 TX logic structure



After TX FIFO ready data, load TX FIFO data to tx_buf0 and tx_buf1.

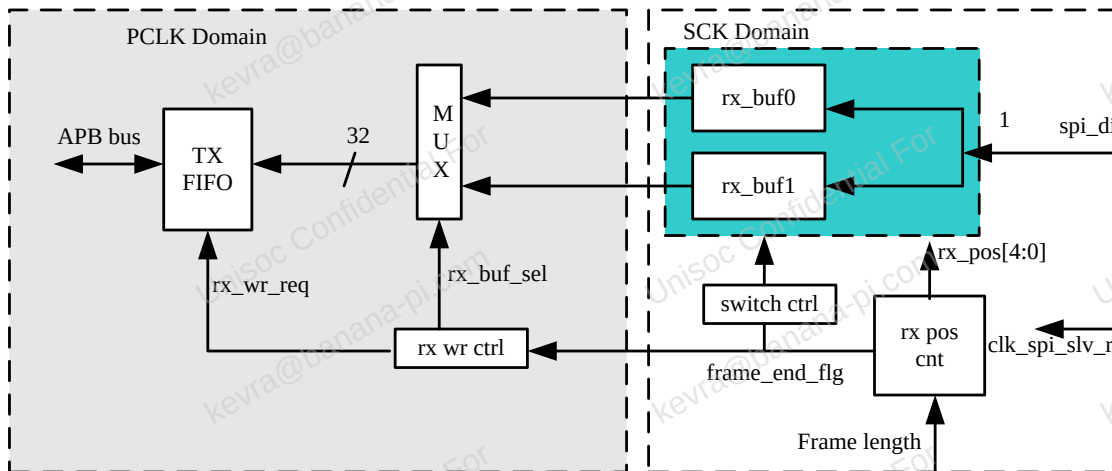
Waiting master send serial clock to slave.

Figure 8-51 TX timing chart



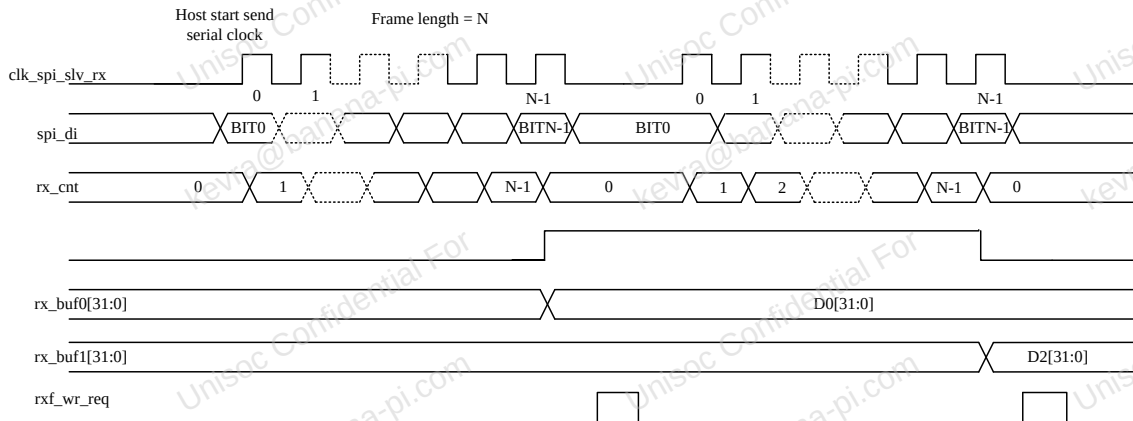
8.12.3.7.4 SPI Slave RX

Figure 8-52 RX Logic structure



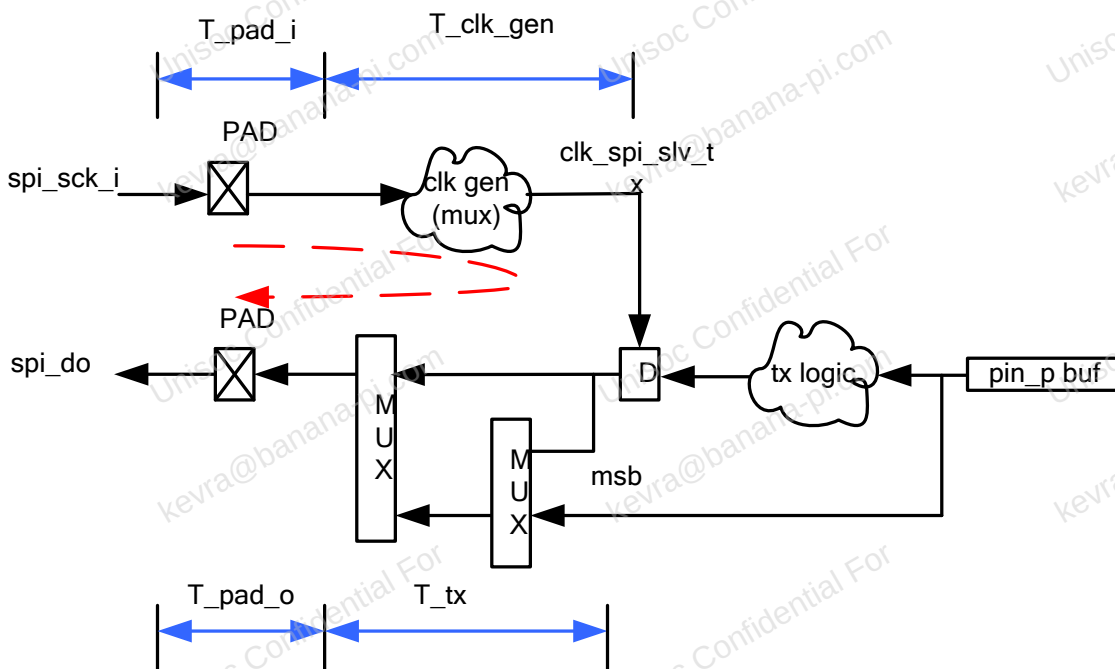
- According to SPI interface timing mode requirements, clock generator generates `clk_spi_slv_rx`.
- Rx pos counter count `clk_spi_slv_rx`, generating `rx_buf_sel` signal during the last clock per frame. If `rx_buf_sel=0`, sample the serial data input to `rx_buf0`, otherwise sample to `rx_buf1`.
- Synchronize the `rx_buf_sel` signal to PCLK domain, get the edge, and generate `rx_wr_req`. fetch data from `rx_buf0`, write to FIFO during the rising edge, and `rx_buf1` during the falling edge.

Figure 8-53 RX timing chart



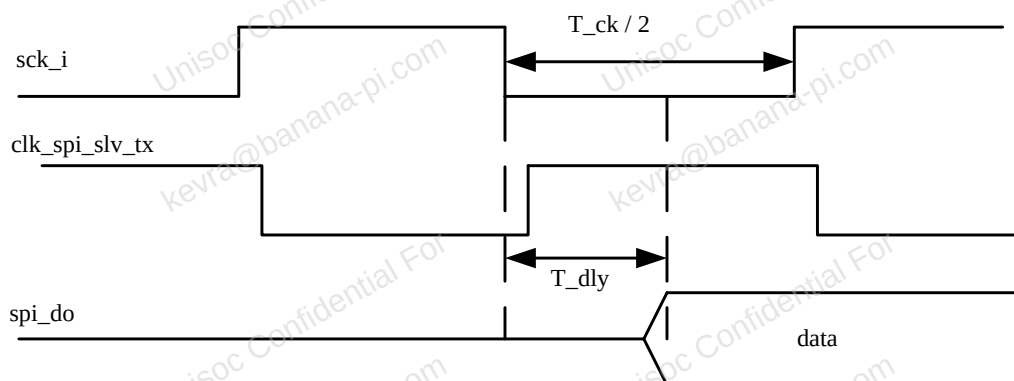
8.12.3.7.5 SPI_HS Slave Clock

Figure 8-54 SPI_HS Slave Clock Path



When TX transmission mode, loop timing delay from sck in to spi_do is : $T_{dly} = T_{pad_i} + T_{clk_gen} + T_{tx} + T_{pad_o}$. according to SPI timing requirement, send data during rising edge of SCK, and sample data during falling edge of SCK. Otherwise, send data during falling edge of SCK, and sampling data during rising edge of SCK, so, timing should satisfied $T_{ck}/2 - T_{dly} > T_{setup}$.

Figure 8-55 Slave Clock Delay Path



8.12.4 Control Registers

See *UIS7885 AP System Control Registers Instructions*.

8.12.5 Application Notes

8.12.5.1 Programming Model

8.12.5.1.1 MICRO/PLUS mode program examples

1. **MICRO/PLUS mode**, master, 32 bits per-channel, MCLK 153.6M, SCK 76.8M, CS0 is valid, RX at negedge, TX at posedge, SCK idle phase being low, Program control register:
 - `Spi_clkd = 0x0001;`
 - `Spi_ctl0 = 0x0e01;`
 - `Spi_ctl1 = 0x3100;`
 - `Spi_ctl2 = 0x0000;`
 - `Spi_ctl3 = 0x1d1a;`
 - `Spi_ctl4 = 0x0000;`
 - `Spi_ctl5 = 0x0000;`
 - `Spi_ctl6 = 0x1303;`
 - `Spi_ctl7 = 0x0000;`
 - `Spi_ctl8 = 0x0000;`
 - `Spi_ctl9 = 0x0019;`
 - `Spi_ctl10 = 0x0000;`
 - `Spi_ctl11 = 0x0019;`
2. 16 bits per-channel, master, MCLK 80M, SCK 10M, only receive mode, ready to receive 0x1f words from slave, CS1 is valid, RX at falling edge, TX at rising edge, SCK idle phase being low, Program control register:
 - `Spi_clkd = 0x0003;`
 - `Spi_ctl0 = 0x0d41;`
 - `Spi_ctl1 = 0x1000;`
 - `Spi_ctl2 = 0x0000;`

- Spi_ctl3 = 0x1010;
 - Spi_ctl4 = 0x021f;
 - Spi_ctl5 = 0x0000;
 - Spi_ctl6 = 0x1010
 - Spi_ctl7 = 0x0000
 - Spi_ctl8 = 0x0000
 - Spi_ctl9 = 0x0011
 - Spi_ctl10 = 0x0000
 - Spi_ctl11 = 0x0011
3. 8 bits per-channel, master, MCLK 48M, SCK 24M, CS2 is valid, RX at falling edge, TX at rising edge, SCK idle phase being high, LSB, dma_en, Program control register:
- Spi_clkd = 0x0000;
 - Spi_ctl0 = 0x27a2;
 - Spi_ctl1 = 0x3000;
 - Spi_ctl2 = 0x0040; //bit[6], enable dma
 - Spi_ctl3 = 0x1010;
 - Spi_ctl4 = 0x0000;
 - Spi_ctl5 = 0x0000;
 - Spi_ctl6 = 0x1010
 - Spi_ctl7 = 0x0000;
 - Spi_ctl8 = 0x0000
 - Spi_ctl9 = 0x0011
 - Spi_ctl10 = 0x0000
 - Spi_ctl11 = 0x0011
4. MICRO/PLUS mode, master, 32 bits per-channel, MCLK 80M, SCK 20M, CS0 is valid, RX at rising edge, TX at negedge, SCK idle phase being low, transmit only, Program control register:
- Spi_clkd = 0x0001;
 - Spi_ctl0 = 0x0e02;
 - Spi_ctl1 = 0x2000; // transmit only
 - Spi_ctl2 = 0x0000;
 - Spi_ctl3 = 0x1010;
 - Spi_ctl4 = 0x0000;
 - Spi_ctl5 = 0x0000;
 - Spi_ctl6 = 0x1010
 - Spi_ctl7 = 0x0000;
 - Spi_ctl8 = 0x0000
 - Spi_ctl9 = 0x0011
 - Spi_ctl10 = 0x0000
 - Spi_ctl11 = 0x0011
5. MICRO/PLUS mode, **slave**, 32 bits per-channel, RX at rising edge, TX at negedge, SCK idle phase being low, Program control register :
- Spi_clkd //no use when SPI working as slave , half SCK must have at least 4 system(MCLK) clock cycle.

- Spi_ctl0 = 0x0f02;
- Spi_ctl1 = 0x3000;
- Spi_ctl2 = 0x0020;
- Spi_ctl3 = 0x1010;
- Spi_ctl4 = 0x0000;
- Spi_ctl5 = 0x0080;
- Spi_ctl6 = 0x1010

6. MICRO/PLUS mode, slave, 8 bits per-channel, RX at falling edge, TX at rising edge, SCK idle phase being low, TX only/ RX only, Program control register:

- Spi_ctl0 = 0x0e21;
- Spi_ctl1 = 0x2000(TX only); / spi_ctl1 = 0x1000(RX only)
- Spi_ctl2 = 0x0020;
- Spi_ctl3 = 0x1010;
- Spi_ctl4 = 0x0000;
- Spi_ctl5 = 0x0090;
- Spi_ctl6 = 0x1010

8.12.5.1.2 SYNC mode program examples

Only work in master mode.

Relative register: spi_ctl0[15:14], spi_ctl1[4:0], spi_ctl1[11:8].

SYNC mode, master, 32 bits per-channel, MCLK 48M, SCK 12M, RX at rising edge, TX at falling edge, CS0 is valid , and signal sync maps to cs2, Program control register:

- Spi_clkd = 0x0001;
- Spi_ctl0 = 0x4a02; //bit[14] enable sync mode, bit[15] = 0 means positive pulse
- Spi_ctl1 = 0x3400; // sync pulse position will locate on top of bit0, sync maps to cs2
- Spi_ctl2 = 0x0000;
- Spi_ctl3 = 0x1010;
- Spi_ctl4 = 0x0000;
- Spi_ctl5 = 0x0000;
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0x0000;
- Spi_ctl8 = 0x0000
- Spi_ctl9 = 0x0011
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0011

8.12.5.1.3 4-WIRE(S8) mode program examples

only work in master mode and only operation write being supported, TX at falling edge relative register:

spi_ctl1[7], spi_ctl1[11:8].

S8 mode, 8 bits per-channel, TX at falling edge, CS0 is valid and signal CD maps to cs1, program control register:

- Spi_ctl0 = 0x2e21;
- Spi_ctl1: 0x3280;

- Spi_ctl2 = 0x0000;
- Spi_ctl3 = 0x1010;
- Spi_ctl4 = 0x0000;
- Spi_ctl5 = 0x0000;
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0x0000;
- Spi_ctl8 = 0x0000
- Spi_ctl9 = 0x0011
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0011

8.12.5.1.4 3-WIRE mode program examples

Only work in master mode.

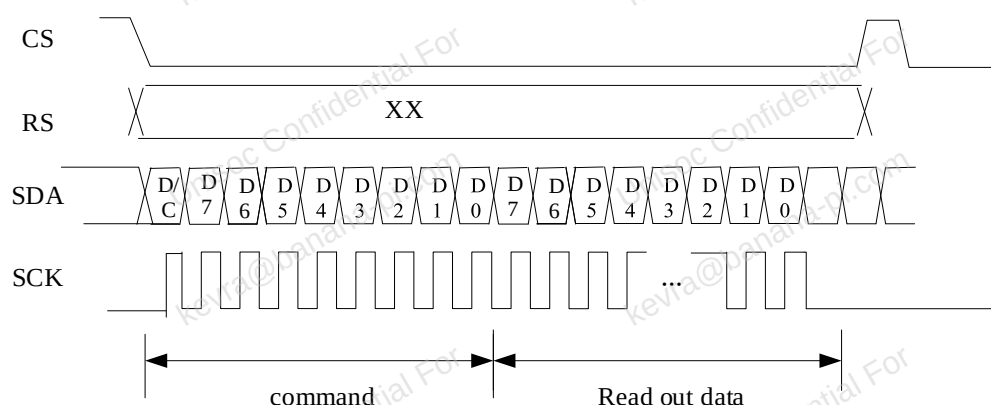
Relative register: spi_ctl0[15], spi_ctl1[4:0], spi_ctl1[5], spi_ctl1[6], spi_ctl2[4:0].

1. timing1(TX RX at falling edge), 16 bits per-channel, CS0 is valid, read command polarity is high, w/r control position is 8, read data start bit is 7, program control register:
 - spi_ctl0 = 0x0e43;
 - spi_ctl1 = 0x3068;
 - spi_ctl2 = 0x0007;
2. timing2(TX RX at rising edge), 16 bits per-channel, CS0 is valid, read command polarity is low, w/r control position is 8, read data start bit is 7, program control register:
 - spi_ctl0 = 0xae43;
 - spi_ctl1 = 0x3068;
 - spi_ctl2 = 0x0007;

8.12.5.2 GC9304C LCD driver Programming Model

8.12.5.2.1 8 bit read program examples

Figure 8-56 8 bit read process timing

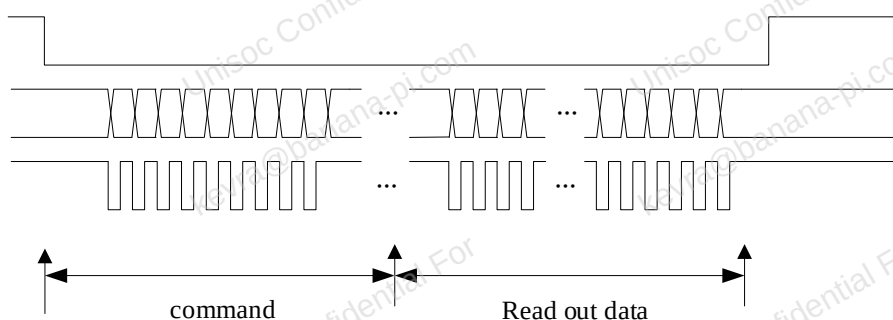


Only work in master mode, TX at falling edge and received at rising edge. Use 9bit 3 wire RW mode. 1 command bit, 8 bits output data and 8 bit input data per-channel, no dummy cycle.

- Spi_ctl0 = 0x2e41
- Spi_ctl1 = 0x3100
- Spi_ctl2 = 0x0000
- Spi_ctl3 = 0x1010
- Spi_ctl4 = 0x0000
- Spi_ctl5 = 0x0000
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0x0028 3 wire 9bit RW mode
- Spi_ctl8 = 0x0000
- Spi_ctl9 = 0x0011
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0011
- Spi_ctl12 = 0x0002

8.12.5.2.2 24/32 bit read program examples

Figure 8-57 24/32 bit read process timing



Only work in master mode, TX at falling edge and received at rising edge. Use 9bit 3 wire mode and 4 wire 8bit mode. The read transfer divided into 2 transactions. The first one send 1 command bit and 8 bits output command data, the second transaction received 24/32 bit read data use 4 wire 8bit mode.

Figure 8-58 2 transactions of read transfer process



1. 3 wire 9 bit mode transmit command
 - Spi_ctl0 = 0x2e21
 - Spi_ctl1 = 0x2100
 - Spi_ctl2 = 0x0007
 - Spi_ctl3 = 0x1010
 - Spi_ctl4 = 0x0000

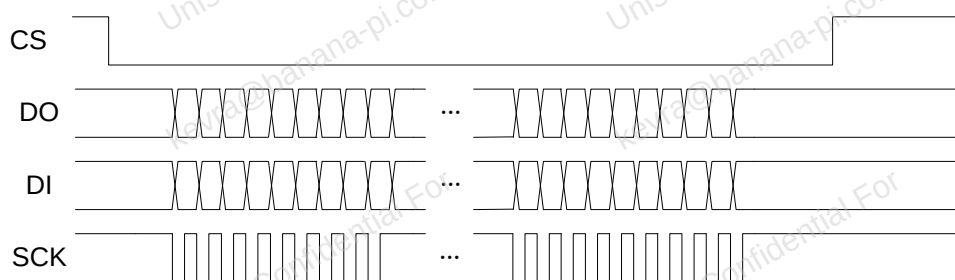
- Spi_ctl5 = 0x0000
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0x0008 3 wire 9 bit mode
- Spi_ctl8 = 0x0010 TX dummy length=1
- Spi_ctl9 = 0x0001
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0001
- Spi_ctl12 = 0x0002

2. Read data in 4 wire 8bit mode

- Spi_ctl0 = 0x2e01 32bit data
- Spi_ctl1 = 0x1100
- Spi_ctl2 = 0x0007
- Spi_ctl3 = 0x1010
- Spi_ctl4 = 0x0000
- Spi_ctl5 = 0x0000
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0x0018 4 wire 8 bit mode
- Spi_ctl8 = 0x0000
- Spi_ctl9 = 0x0001
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0001
- Spi_ctl12 = 0x0001

8.12.5.2.3 Two data line mode program examples

Figure 8-59 two data line mode timing

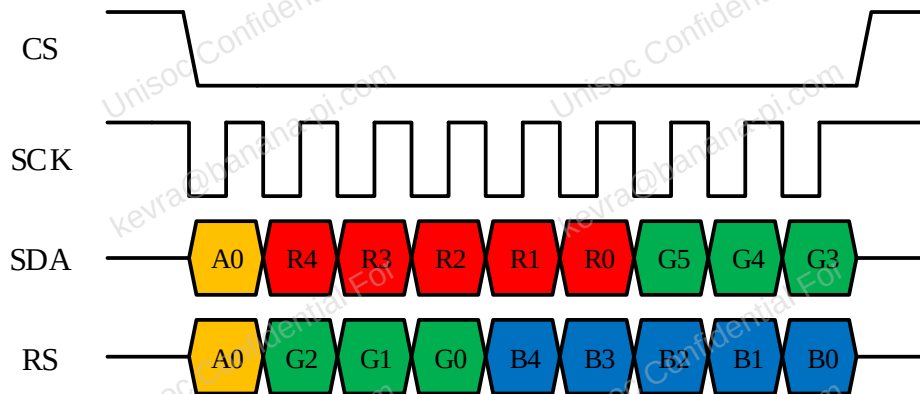


Only work in master mode, TX at falling edge and received at rising edge. Use 9bit 3 wire mode send RGB data. In this mode, DI pin is second data line.

1. 3 wire 9 bit mode transmit RGB data (RGB565 1pixel mode)

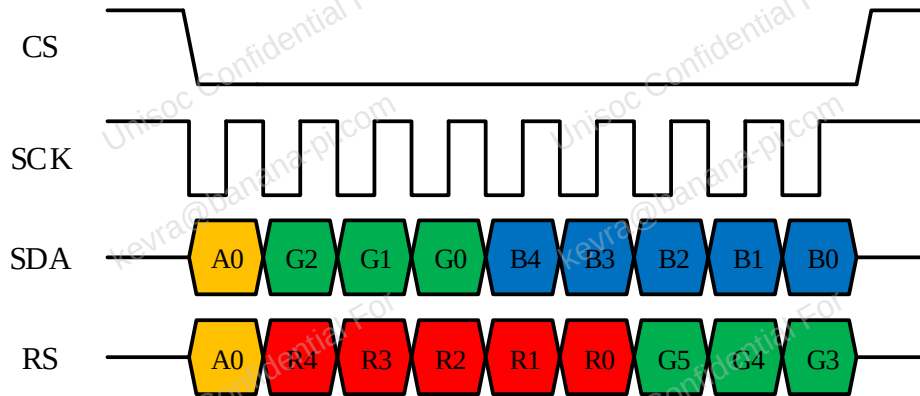
- Spi_ctl0 = 0x2e21
- Spi_ctl1 = 0x2100
- Spi_ctl2 = 0x0007
- Spi_ctl3 = 0x1010
- Spi_ctl4 = 0x0000

- Spi_ctl5 = 0x0000
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0xc008 3 wire 9 bit mode, enable two data line, data format RGB565
- Spi_ctl8 = 0x0000
- Spi_ctl9 = 0x0001
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0001
- Spi_ctl12 = 0x0002



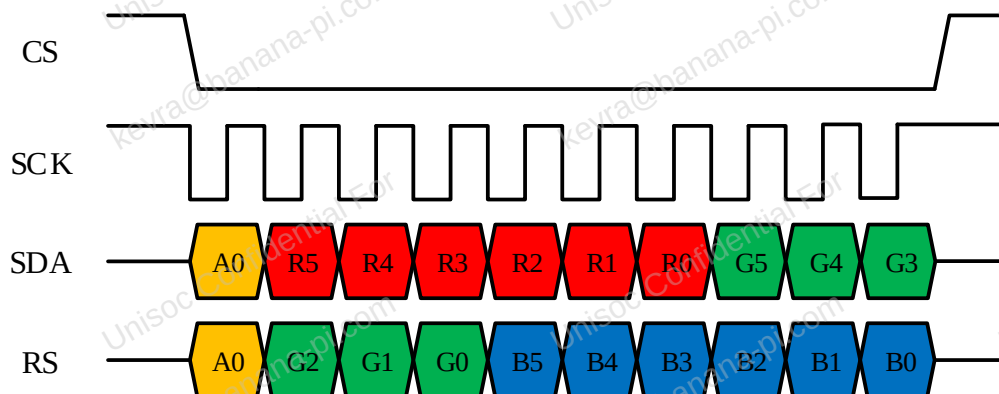
2. 2 data line switch :3 wire 9 bit mode transmit RGB data (RGB565 1pixel mode)

- Spi_ctl0 = 0x2e21
- Spi_ctl1 = 0x2100
- Spi_ctl2 = 0x0007
- Spi_ctl3 = 0x1010
- Spi_ctl4 = 0x0000
- Spi_ctl5 = 0x0000
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0xc008 3 wire 9 bit mode, enable two data line, data format RGB565
- Spi_ctl8 = 0x0800 data_line2_sw = 1
- Spi_ctl9 = 0x0001
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0001
- Spi_ctl12 = 0x0002



3. 3 wire 9 bit mode transmit RGB data (RGB666 1pixel mode)

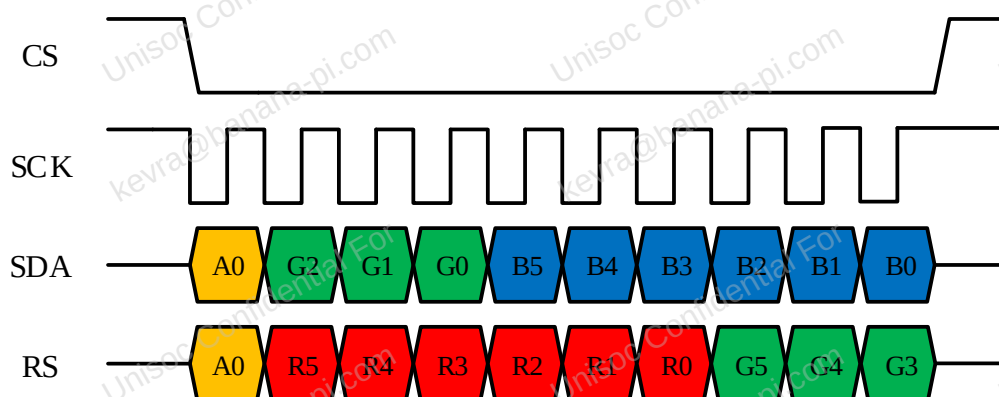
- Spi_ctl0 = 0x2e21
- Spi_ctl1 = 0x2100
- Spi_ctl2 = 0x0007
- Spi_ctl3 = 0x1010
- Spi_ctl4 = 0x0000
- Spi_ctl5 = 0x0000
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0xa008 3 wire 9 bit mode, enable two data line, data format RGB666
- Spi_ctl8 = 0x0000
- Spi_ctl9 = 0x0001
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0001
- Spi_ctl12 = 0x0002



4. 2 data line switch: 3 wire 9 bit mode transmit RGB data (RGB666 1pixel mode)

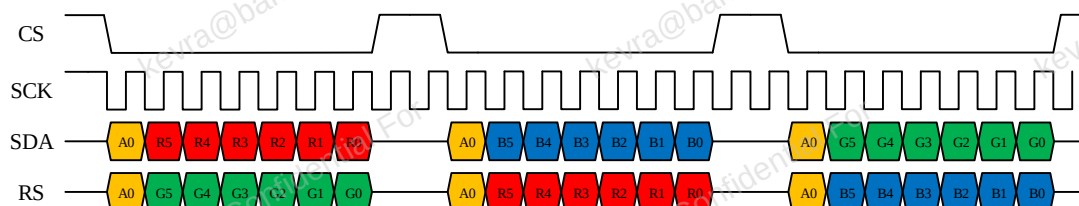
- Spi_ctl0 = 0x2e21
- Spi_ctl1 = 0x2100
- Spi_ctl2 = 0x0007
- Spi_ctl3 = 0x1010

- Spi_ctl4 = 0x0000
- Spi_ctl5 = 0x0000
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0xa008 3 wire 9 bit mode, enable two data line, data format RGB666
- Spi_ctl8 = 0x0800 data_line2_sw = 1
- Spi_ctl9 = 0x0001
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0001
- Spi_ctl12 = 0x0002



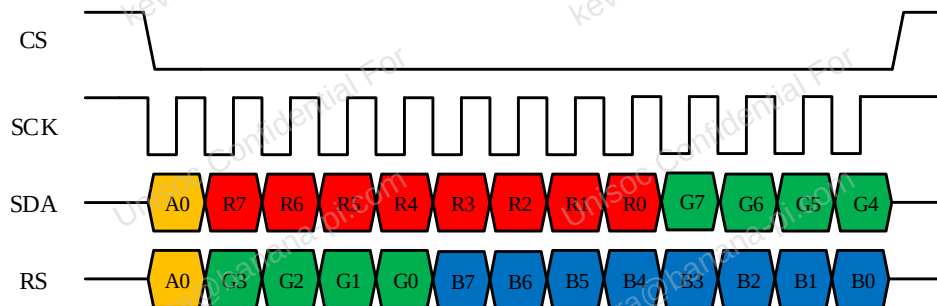
5. 3 wire 9 bit mode transmit RGB data (RGB666 2/3 pixel mode)

- Spi_ctl0 = 0x2e21
- Spi_ctl1 = 0x2100
- Spi_ctl2 = 0x0007
- Spi_ctl3 = 0x1010
- Spi_ctl4 = 0x0000
- Spi_ctl5 = 0x0000
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0xa008 3 wire 9 bit mode, enable two data line, data format RGB666
- Spi_ctl8 = 0x1000 2/3 pixel mode
- Spi_ctl9 = 0x0001
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0001
- Spi_ctl12 = 0x0002



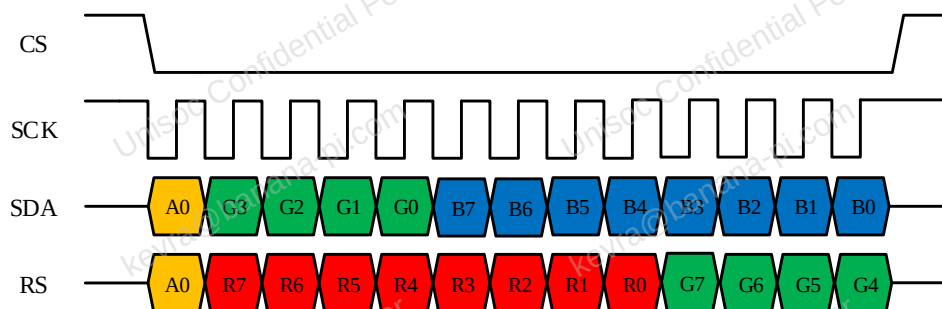
6. 3 wire 9 bit mode transmit RGB data (RGB888 1 pixel mode)

- Spi_ctl0 = 0x2e21
- Spi_ctl1 = 0x2100
- Spi_ctl2 = 0x0007
- Spi_ctl3 = 0x1010
- Spi_ctl4 = 0x0000
- Spi_ctl5 = 0x0000
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0x9008 3 wire 9 bit mode, enable two data line, data format RGB888
- Spi_ctl8 = 0x0000
- Spi_ctl9 = 0x0001
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0001
- Spi_ctl12 = 0x0002



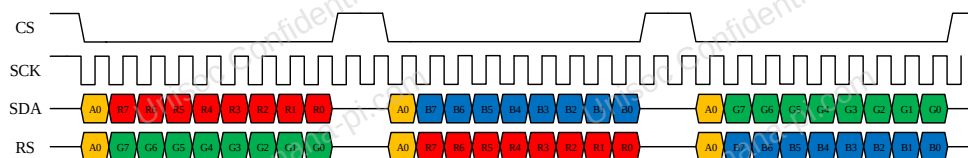
7. 2 data line switch: 3 wire 9 bit mode transmit RGB data (RGB888 1pixel mode)

- Spi_ctl0 = 0x2e21
- Spi_ctl1 = 0x2100
- Spi_ctl2 = 0x0007
- Spi_ctl3 = 0x1010
- Spi_ctl4 = 0x0000
- Spi_ctl5 = 0x0000
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0x9008 3 wire 9 bit mode, enable two data line, data format RGB888
- Spi_ctl8 = 0x0800 data_line2_sw = 1
- Spi_ctl9 = 0x0001
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0001
- Spi_ctl12 = 0x0002



8. 3 wire 9 bit mode transmit RGB data (RGB888 2/3pixel mode)

- Spi_ctl0 = 0x2e21
- Spi_ctl1 = 0x2100
- Spi_ctl2 = 0x0007
- Spi_ctl3 = 0x1010
- Spi_ctl4 = 0x0000
- Spi_ctl5 = 0x0000
- Spi_ctl6 = 0x1010
- Spi_ctl7 = 0x9008 3 wire 9 bit mode, enable two data line, data format RGB888
- Spi_ctl8 = 0x1000 2/3 pixel mode
- Spi_ctl9 = 0x0001
- Spi_ctl10 = 0x0000
- Spi_ctl11 = 0x0001
- Spi_ctl12 = 0x0002



8.12.5.3 Programming Notes

1. only receive mode

- as master: set spi_ctl4[9] = 1 and set spi_ctl1[13:12] = "01" and program spi_ctl4[8:0] to N, means enable rx only mode, and ready to receive N words from slave.
- as slave: set spi_ctl1[13:12] = "01", the received number depending on master the posedge of is_rx_only(spi_ctl4[9] = 1) will trigger the process, so SW should write to "0" first, then write to "1" to this bit.

2. fast mode

used for slave mode. In slave mode, half SCK must have at least 4 system clock cycle.

When SCK < 14X MCLK, recommend to use fast mode.

- 8X: is_fst(spi_ctl4[14]) = 1, phs_dly([spi_ctl4[13:12]]) = 1
- 10X: is_fst=1, phs_dly = 2
- 12X: IS_fst=1, phs_dly = 3

3. In salve mode, $\text{spi_ctl5}[15:0] \times \text{Tclk_spi}$ should not be such smaller than Tspi_sck , so as not to make SPI stop transfer.
4. DMA enable
 - a DMA should be configured
 - b Program SPI control register except $\text{spi_ctl2}[6]$
 - c Write $\text{spi_ctl2}[6]$
5. When change SPI control register configuration, SW should ensure:
 - a tx_fifo is real empty
 - b rx_fifo is real empty
 - c change SPI control register

8.13 IIS

8.13.1 Overview

IIS is a common digital audio interface specification. Base Band Chip IIS Interface can be used to implement a CODEC interface with external digital audio system working as host or slave. The IIS interface supports both IIS and PCM data format. The interface can transmit and receive data simultaneously as well as transmit or receive data alternatively at a time. This document defines function and configuration of IIS block.

IIS module can be controlled both by Arm® processor or DSP.

8.13.2 Features

- Compliant with the IIS/PCM standard
- Support IIS, MSB-Justified and DSP data bus interface working on IIS mode
- Support Long Frame, Short Frame and Multi-cycle Frame working on PCM mode
- Serial clock with programmable frequency
- Two 32-word-deep FIFOs minimizes processor overhead at high data rates
- LRCK polarity can be configured
- Wide selection of IIS data lengths 8,16,32bits
- Support master / slave mode
- Support MSB / LSB mode
- Transmit only/ receive only / transmit and receive mode
- 2 DMA requests
- Single interrupt line for multiple interrupt source events
- IIS and PCM master mode: support integer clock divider
- IIS master mode: support fraction clock divider – both variable and fixed serial clock mode
- PCM master mode: support fraction clock divider – only variable serial clock mode
- Support 32bit width of APB 2.0 configuration.
- Support multi data channels(parameter CHN_NUM controls only support 1 & 2)
- Support tx & rx FIFO depth configurable (parameters TX_FIFO_DEPTH & RX_FIFO_DEPTH only support 5/6/7/8 which means FIFO depth 32/64/128/256)
- Support RX only feature for master mode

- Support max 4 data channels
- Support IIS mode & slave mode tx_ng=1 to improve slave mode timing

8.13.3 Function Description

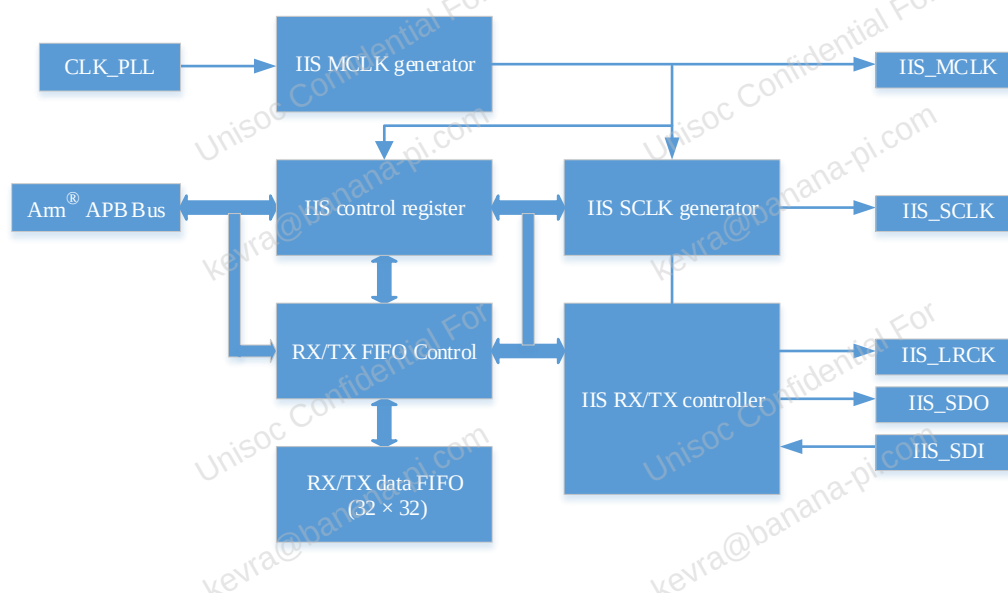
8.13.3.1 IIS interface

IIS module consists of RX/TX controller, Arm® processor APB interface control register, clock generator and FIFO control. Arm® program IIS register, enable IIS mode, starts RX/TX data transfer. Output data is written to TX FIFO and received data is stored in RX FIFO. Each FIFO is 32-bit width plus 32-address length.

MCLK generator generates main clock for IIS block. This clock also output as master clock during hand shaking. In slave mode, external clock source can be input, working as the IIS main clock. Clock divider value and control signal for MCLK come from global control register.

SCLK generator divides main clock according to a 16-bit clock divider register, which defines half SCLK clock width. It can be programmed from 0 to 32767.

Figure 8-60 IIS module diagram

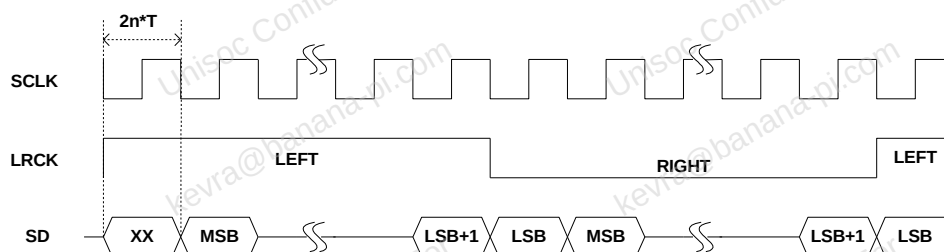
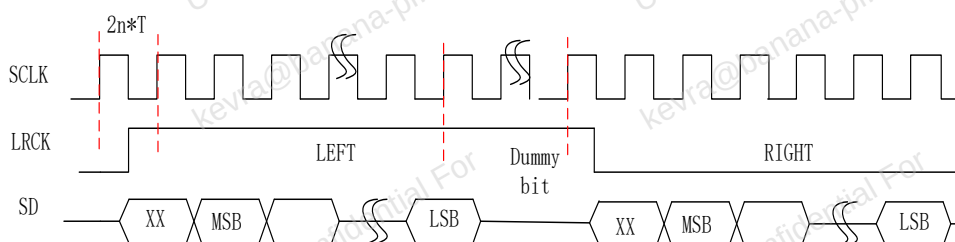


IIS interface has 4 pins, IIS_SCLK is serial clock output, IIS_SDI is serial data input, IIS_SDO is serial data output. IIS_LRCK is left/right channel select. The device generating IIS_LRCK and IIS_SCLK is the master. IIS_LRCK and IIS_SCLK are output pins if working as master. They are input pins if working as slave.

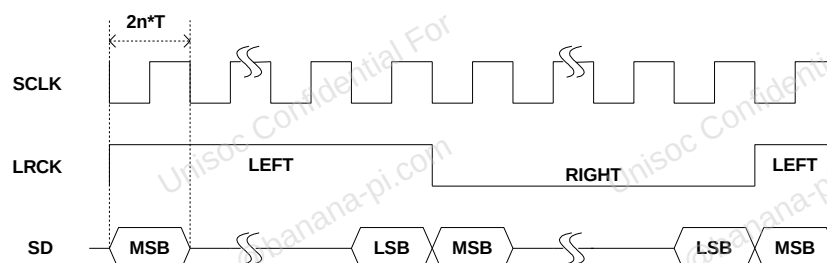
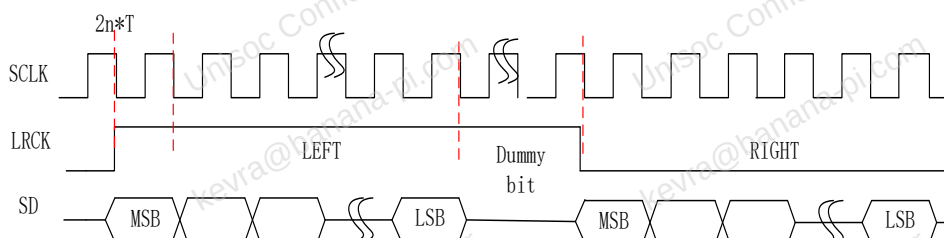
Serial data is transmitted in two's complement format with the MSB first. Serial data can be sent at rising or falling edge of SCLK, at receiver, it must be latched at rising edge. When system word length is greater than the transmitter word length, the word is truncated (least significant data bits are set to '0') for data transmission. If the receiver is sent more bits than its word length, the bits after the LSB are ignored. On the other hand, if the receiver is sent fewer bits than its word length, the missing bits are set to zero internally.

Three bus formats are available in this module. Following Figures illustrate the relationship between the SCLK, LRCK and serial data I/O for different interface protocols. The polarity of LRCK is programmable. The bits per-channel is programmable up to 32bit.

IIS mode is where the MSB is available on the 2nd rising edge of SCLK following a LRCK transition.

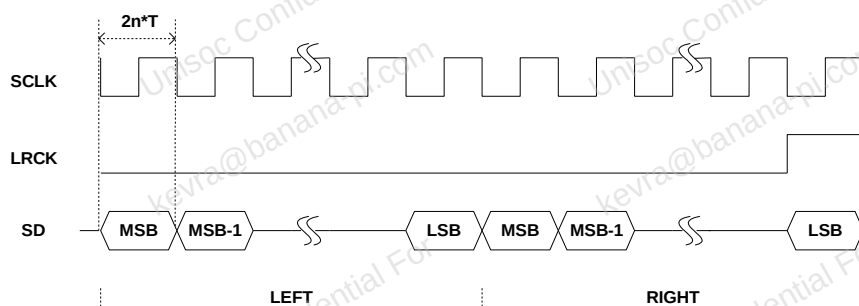
Figure 8-61 IIS-compatible format

Figure 8-62 IIS-compatible format with dummy bit


MSB-justified mode is where the MSB is available on the first rising edge of SCLK following a LRCK transition.

Figure 8-63 MSB-justified format

Figure 8-64 MSB-justified format with dummy bit


DSP mode is where the left channel MSB is available on either the 1st or 2nd rising edge of SCLK following a LRCK transition high. Right channel data immediately follows left channel data.

Figure 8-65 Synchronized format



NOTE

- T : $1/(\text{system clock frequency})$
- n : register programmable integer, $n-1 = 0, \dots, 32767$

Table 8-13 IIS data format memory map

IIS_CTL bit1~0	Bit / channel	Data arrangement
00	8bit	Data on bit 31~24
01	16bit	Data on bit 31~15
1x	32bit	Data on bit 31~0

8.13.3.2 PCM interface

PCM interface is a direct voice interface connects to standard CODEC. The implementation is compliant with the MP-PCM requirements for voice transfer (8 kHz PCM_SYNC and 8 or 16 bits data). The four signals of the PCM interface are:

- PCM_CLK: PCM clock
- PCM_SYNC: PCM 8kHz synchronization signal
- PCM_OUT: PCM output data
- PCM_IN: PCM input data

These four pins is able to share IIS interface follow this mapping:

- PCM_CLK share with IIS_SCLK
- PCM_IN share with IIS_SDI
- PCM_OUT share with IIS_SDO
- PCM_SYNC share with IIS_LRCK

The data can be linear PCM (13-16 bit), μ -Law (8 bit) or A-Law (8bit). The interface can work as either Master or Slave.

Programming IIS_CTL 0 register bit 15 to 1 configures the PCM mode.

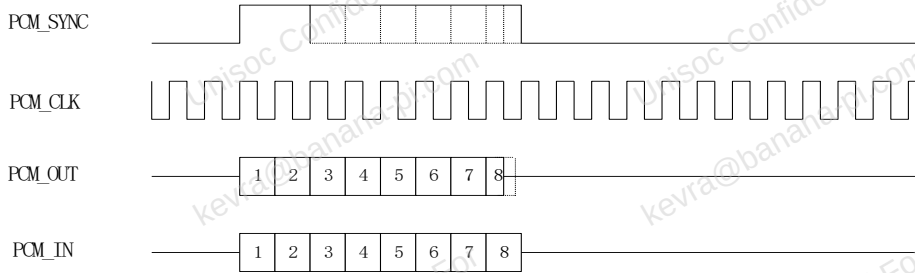
Long Frame Sync is the name given to a clocking format that controls the transfer of PCM data words or samples. In Long Frame Sync, the rising edge of PCM_SYNC indicates the start of the PCM word. When Base Band Chip

is configured as PCM Master, generating PCM_SYNC and PCM_CLK, 8 bits in one frame, then PCM_SYNC is 7-bit long. When Base Band chip is configured as PCM Slave, PCM_SYNC may be from one consecutive falling edges of PCM_CLK to not big or equal than PCM data length (as figure shows, PCM_SYNC is 2~7.5 bits) long.

Programming IIS_CTL0 register bit8 to zero configures the Long Frame Sync mode.

PCM_IN is captured on the falling edge of PCM_CLK and PCM_OUT transmits on the rising edge. PCM_OUT may be configured to be high impedance on the falling edge of PCM_CLK in the LSB position or on the rising edge. The control bit is IIS_CTL2 bit10, if high, OE changes to low at falling edge of the last bit.

Figure 8-66 Long frame sync (Shown with 8-bit sample)

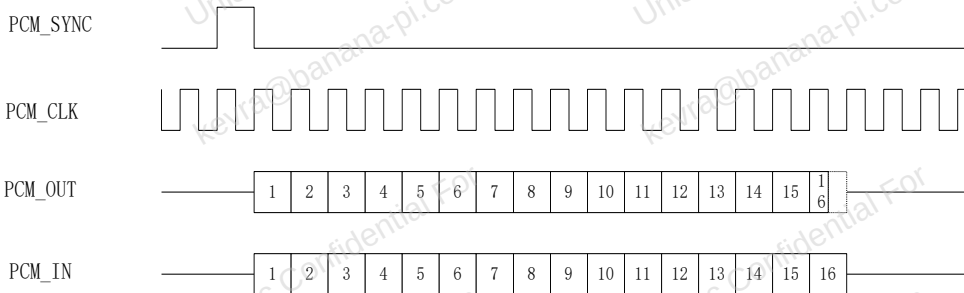


In Short Frame Sync the falling edge of PCM_SYNC indicates the start of the PCM word. PCM_SYNC is always one clock cycle long.

As with Long Frame Sync, Base Band Chip samples PCM_IN on the falling edge of PCM_CLK and transmits PCM_OUT on the rising edge. PCM_OUT may be configured to be high impedance on the falling edge of PCM_CLK in the LSB position or on the rising edge.

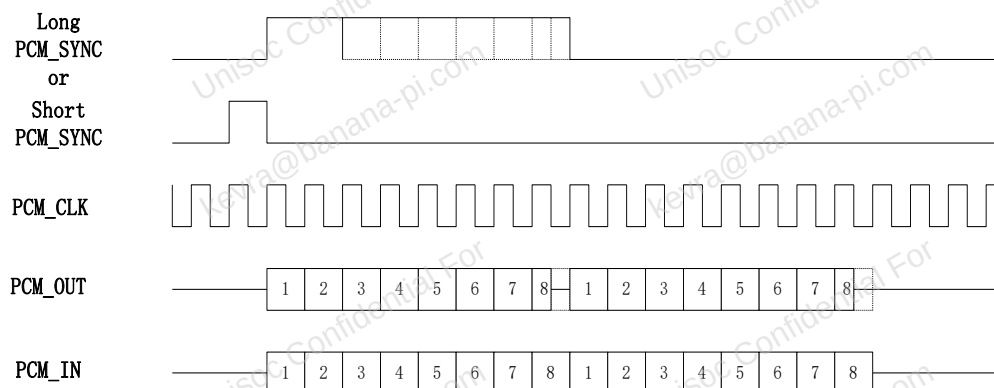
Programming IIS_CTL0 register bit8 to high configures the Short Frame Sync mode.

Figure 8-67 Short frame sync (Shown with 16-bit sample)



More than one channel connection over the PCM interface is supported using multiple slots. Up to three channel connections can be carried over any of the first three slots.

Figure 8-68 Multi-slot operation with two slots and 8-bit compounded samples



Three slots can be configured in one PCM frame; IIS_CTL2 register bit 2~0 define which channel is occupied. Each bit configures one of the three channels. If bit0 is high, then slot0 is carrying data. If channel 0 and 2 is active, this register is 0x5.

PCM data format in TX/RX buffer is different between master and slave mode. If burst size is 16 or 8bit, in slave mode, data for slot 0 and slot 1 are packed into one word, which locates in lower address; data for slot2 locates in higher address; slot0 and slot2 fill higher bits (bit31~24). In master mode, only slot0 takes lower address and locates on lower bits (bit15~8); slot1, slot2 packed in higher address, and slot2 takes lower bits while slot1 in higher bits. Following table shows PCM data format for master and slave mode.

Table 8-14 PCM data format for master mode

IIS_CTL bit1~0	Bit / channel	Data arrangement
00	8bit	Data for channel 0 on bit 15~8, in lower address. Data for channel 1 on bit 31~24, Data for channel 2 on bit 15~8, on higher address.
01	16bit	Data for channel 0 on bit 15~0, in lower address. Data for channel 1 on bit 31~16, Data for channel 2 on bit 15~0, on higher address
1x	32bit	Data for channel 0 store in low address, data for channel 1, 2 store in high address.

Table 8-15 PCM data format for slave mode

IIS_CTL bit1~0	Bit / channel	Data arrangement
00	8bit	Data for channel 0 on bit 31~24, Data for channel 1 on bit 15~8, in lower address. Data for channel 2 on bit 31~24, on higher address.

IIS_CTL bit1~0	Bit / channel	Data arrangement
01	16bit	Data for channel 0 on bit 31~16, Data for channel 1 on bit 15~0, in lower address. Data for channel 2 on bit 31~16, on higher address
1x	32bit	Data for channel 0 store in low address, data for channel 1, 2 store in high address.

Table 8-16 PCM multiple slot and PCM cycle configuration suggestion

pcm_slot[0]	pcm_slot[1]	pcm_slot[2]	pcm_cycle
0	0	1	<=2
0	1	0	>=1
0	1	1	>=1
1	0	0	>=2
1	0	1	>=2
1	1	0	>=2
1	1	1	>=2

Table 8-17 Exceptions for PCM multiple slot configuration

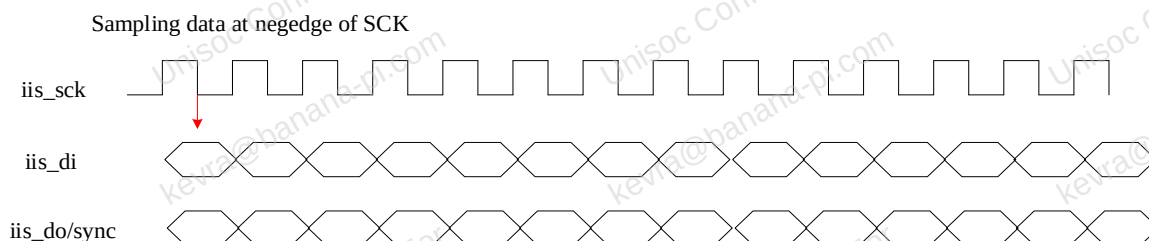
Bit width	pcm_slot[0]	pcm_slot[1]	pcm_slot[2]	pcm_cycle	sample rate	clk_iis
32	1	-	1	2	8 or 44.1 or 48 kHz	26/80MHz
16,8				4		

NOTE

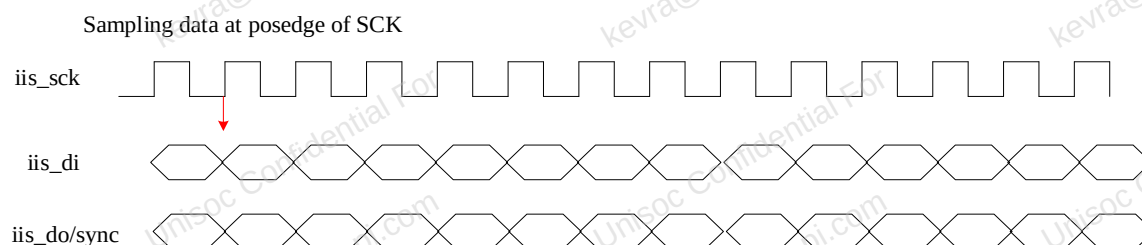
- In PCM fraction divide mode, clk_iis frequency should be 20 times bigger than bit clock
- For EDA simulation, pcm_slot[2:0]=101, bit_width=8 or 16, pcm_cycle>=2 is ok, but software test meet some problem when sample rate > 1kHz. So suggest software engineer setting pcm_cycle=4 in application. It's ok for both software and EDA simulation.

8.13.3.3 Pad IO timing

In default mode, IIS output signals, like iis_do, iis_sync(for master) signal, are sent at the posedge of iis_sck. And input signals like iis_di, iis_sync(for slave), are sampled at the negedge of iis_sck.

Figure 8-69 Sampling data at negedge of IIS_SCK


You can also set output signal sent edge as negedge and sampling edge as posedge, like the following.

Figure 8-70 Sampling data at posedge of IIS_SCK


The implementation of sampling edge is as following.

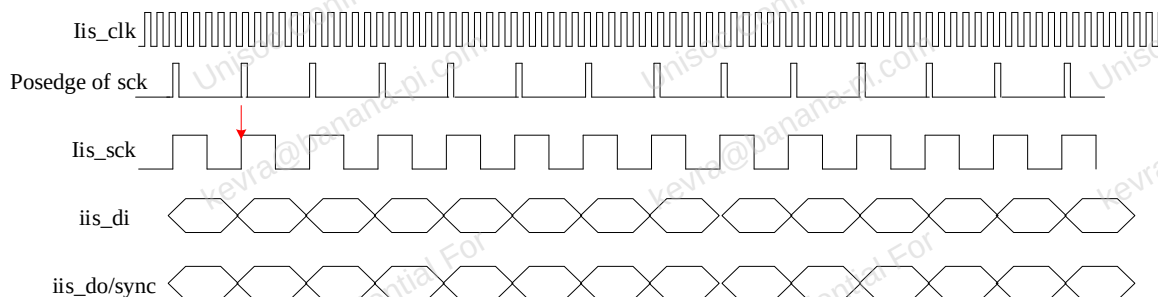
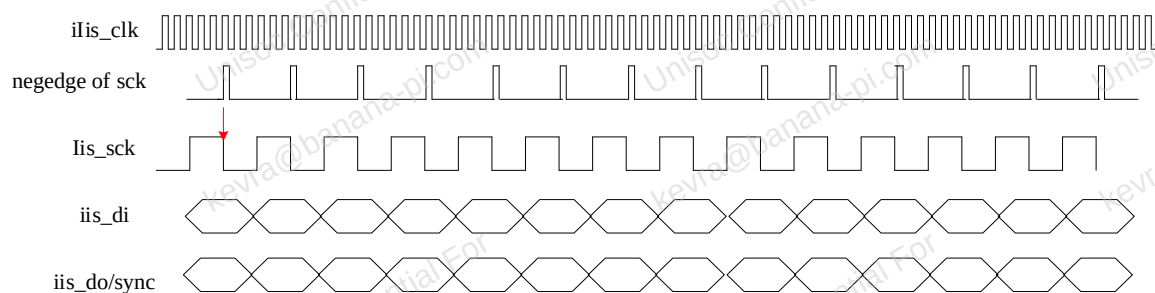
Figure 8-71 implementation of sampling data at negedge of IIS_SCK


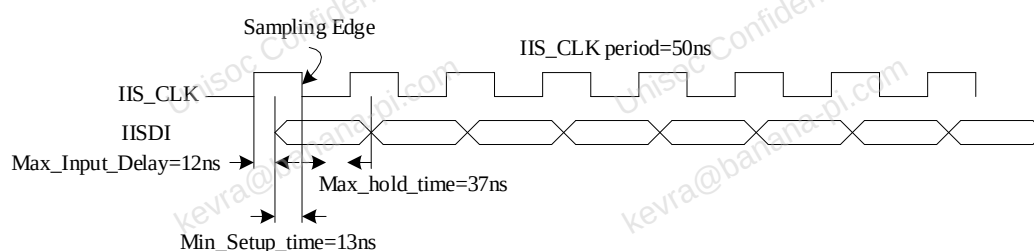
Figure 8-72 implementation of sampling data at posedge of IIS_SCK



8.13.3.4 Setup time and hold time

If assume max input delay is 10ns and min input delay is 0ns for iis_di to iis_sck. max output delay is 10ns and min output delay is 5ns for iis_do/iis_lrck to iis_sck. The clock period time is 50ns for iis_sck.

Figure 8-73 implementation of sampling data at posedge of IIS_SCK



Then for iis_di, the setup time range is $13\text{ns} < \text{setup_time} < 25\text{ns}$, and the hold time range is $25\text{ns} < \text{hold_time} < 37\text{ns}$. For iis_do/iis_lrck, the setup time range is $13\text{ns} < \text{setup_time} < 25\text{ns}$, and the hold time range is $25\text{ns} < \text{hold_time} < 37\text{ns}$.

8.13.4 Control Registers

See *UIS7885 AP System Control Registers Instructions*.

8.13.5 Application Notes

8.13.5.1 IIS Program examples

- Justified format, master, 16 bits per-channel, LRCK low for left, Program control register:
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_08d1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0000
- IIS format, master, 8 bits per-channel, LRCK high for left, Program control register:
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_0dc1
 - iis_ctl1 = 32'h0000_0f11

- iis_ctl2 = 32'h0000_0000
- DSP (sync) format, master, 16 bits per-channel, MSB is available on the 1st rising edge, Program control register
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_0ad1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0000
- DSP format, master, 16 bits per-channel, MSB is available on the 1st rising edge , Program control register:
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_0ed1
 - iis_ctl1 = 32'h0000_3f11
 - iis_ctl2 = 32'h0000_7800
- Justified mode, slave, 8 bits per-channel, enable clock counter, Program control register:
 - iis_ctl0 = 32'h0000_08c9
 - iis_ctl1 = 32'h0000_3f22
 - iis_ctl2 = 32'h0000_0000

8.13.5.2 PCM program examples

Setting IIS_CTL0 register bit[15] to “1” enables IIS module working in PCM mode.

- Long Frame format, master, 8 bits per-channel, RX at falling edge, TX at rising edge, Program control register
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_80c1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0001
- Short Frame format, master, 8 bits per-channel
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_81c1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0001
- Multi-cycle format, master, 8 bits per-channel, pcm_oe=1,pcm_cyc=1, slot0 is occupied
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_80c1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0409
- Multi-cycle format, master, 8 bits per-channel, pcm_cyc=2, slot0 and slot1 are occupied
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_80c1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0013
- Long frame format, slave, 16 bits per-channel, enable clock count
 - iis_ctl0 = 32'h0000_80c9

- iis_ctl1 = 32'h0000_3f11
- iis_ctl2 = 32'h0000_0001
- Short frame format , slave, 16 bits per-channel, enable clock count
 - iis_ctl0 = 32'h0000_81cd
 - iis_ctl1 = 32'h0000_3a1
 - iis_ctl2 = 32'h0000_0001

8.13.5.3 Clock dividing modes

In master mode, the IIS serial clock iis_sck is sent from the IIS module, and this clock is divided from clk_iis. IIS module provides three modes for clock dividing.

- Integer clock dividing mode
 - Both IIS and PCM master support this mode.
 - This mode can be used only if iis_sck is integrally divided from clk_iis.
 - Mode enable: IIS_CLK_DVDR_MD0 = 0 and IIS_CLK_DVDR_MD1 = 0
 - Only IIS_CLKD is used for dividing; IIS_CLKM and IIS_CLKN are DON'T CARE.
 - $IIS_CLKD = (clk_iis / (2 \times iis_sck)) - 1$; $iis_sck = \text{sample rate} \times \text{channel number} \times \text{channel length}$

For example, if clk_iis runs at 153.6MHz, and IIS module is configured as 48k sample rate, 2 channel and 16-bit channel length:

 - iis_sck should be $48k \times 2 \times 16 = 1.536MHz$
 - IIS_CLKD should be $(153.6M / (2 \times 1.536M)) - 1 = 49$
- Fraction clock dividing mode0
 - Both IIS and PCM master support this mode.
 - In this mode, the actual frequency of iis_sck is variable, and only average is target frequency. Because of this variability, the iis_lrck frequency is also variable, and only average is target sample rate.
 - Mode enable: IIS_CLK_DVDR_MD0 = 1 and IIS_CLK_DVDR_MD1 = 0
 - IIS_CLKM and CLKN are used for dividing; while IIS_CLKD is not cared.
 - $IIS_CLKM / IIS_CLKN = 2 \times (\text{target } iis_sck) / clk_iis$; $(\text{target } iis_sck) = \text{sample rate} \times \text{channel number} \times \text{channel length}$
 - $IIS_CLKM = 2 \times \text{sample rate (Hz)} \times \text{channel number} \times \text{channel length} / 100$
 - $IIS_CLKN = clk_iis \text{ (Hz)} / 100$

For example, if clk_iis runs at 153.6MHz, and IIS module is configured as 44.1kHz sample rate, 2 channel and 16-bit channel length:

 - $IIS_CLKM = 2 \times 44100 \times 2 \times 16 / 100 = 28224$
 - $IIS_CLKN = 153600000 / 100 = 1536000$
- Fraction clock dividing mode1
 - Only IIS master supports this mode.
 - In this mode, the frequency of iis_sck is fixed. But the actual sample rate is variable, and only average is target sample rate.
 - Compared with MD0, the advantage of MD1 is that iis_sck is fixed, but the disadvantage is that the jitter of sample rate is higher.
 - Mode enable: IIS_CLK_DVDR_MD0 = 0 and IIS_CLK_DVDR_MD1 = 1
 - IIS_CLKM, CLKN and CLKD are used for dividing
 - $IIS_CLKD = (clk_iis / (2 \times iis_sck)) - 1$

- $IIS_CLKM / IIS_CLKN = (\text{target sample rate}) \times \text{channel number} \times \text{channel length} / iis_sck$; IIS_CLKM must be not more than IIS_CLKN .
- $IIS_CLKM = 2 \times (IIS_CLKD + 1) \times (\text{target sample rate}) \times \text{channel number} \times \text{channel length} / 100$
- $IIS_CLKN = clk_iis / 100$
- The iis_sck (or IIS_CLKD) should be decided by software. iis_sck must be not lower than $((\text{target sample rate}) \times \text{channel number} \times \text{channel length})$. And iis_sck should be not too high for timing issue. Usually iis_sck is a little bit higher than $((\text{target sample rate}) \times \text{channel number} \times \text{channel length})$.

For example, if clk_iis runs at 153.6MHz, and IIS module is configured as 44.1kHz sample rate, 2 channel and 16-bit channel length. Firstly, choose iis_sck as $153.6M / (2 \times (44 + 1)) = 1.706667\text{MHz}$ because $((\text{target sample rate}) \times \text{channel number} \times \text{channel length}) = 1.4112\text{MHz}$ and 1.706667MHz is a little bit higher than 1.4112MHz.

- $IIS_CLKD = 44$
- $IIS_CLKM = 2 \times (44 + 1) \times 44100 \times 2 \times 16 / 100 = 1270080$
- $IIS_CLKN = 153600000 / 100 = 1536000$

Some guidelines for clock dividing modes:

- Integer clock dividing mode is preferred if possible, because this mode has fixed serial clock frequency and fixed sample rate.
- If iis_sck cannot be integrally divided from clk_iis , choose fraction clock dividing modes. If system needs fixed serial clock frequency, mode1 is chosen. Otherwise mode0 is preferred for small sample rate jitter.
- IIS master supports both mode0 and mode1, and PCM master supports only mode0.
- The frequency options of clk_iis depend on different chips. But 153.6MHz is always preferred because this frequency can be divided integrally to support 48k, 24k, 12k, 32k, 16k and 8k sample rate.
- Fraction clock divide mode1 configure.

Table 8-18 IIS configure for fraction clock divide

MD0, MSB-justified	Master	Slave
IIS_CTL[5]	0	x(don't care)
IIS_CTL[4]	1	x(don't care)
IIS_CTL[1]	x(don't care)	x(don't care)
IIS_CTL[0]	x(don't care)	x(don't care)

MD0, compatible	Master	Slave
IIS_CTL[5]	0	x(don't care)
IIS_CTL[4]	1	x(don't care)
IIS_CTL[1]	x(don't care)	x(don't care)
IIS_CTL[0]	x(don't care)	x(don't care)

MD1, MSB-justified	Master	Slave
IIS_CTL[5]	1	x(don't care)
IIS_CTL[4]	0	x(don't care)
IIS_CTL[1]	x(don't care)	0
IIS_CTL[0]	x(don't care)	1

MD1, compatible	Master	Slave
IIS_CTL[5]	1	x(don't care)
IIS_CTL[4]	0	x(don't care)
IIS_CTL[1]	x(don't care)	1
IIS_CTL[0]	x(don't care)	0

Table 8-19 iis_clk configure suggestion

IIS mode	rate	width	clk_iis
compatible	192k	32	153.6M(Suggest)
compatible	192k	32	128M(Suggest)
compatible	192k	32	80M(Not Suggest)
compatible	192k	32	26M(Not Suggest)
compatible	192k	16	153.6M(Suggest)
compatible	192k	16	128M(Suggest)
compatible	192k	16	80M(Suggest)
compatible	192k	16	26M(Not Suggest)
compatible	192k	8	153.6M
compatible	192k	8	128M
compatible	192k	8	80M
compatible	192k	8	26M
compatible	44.1k	32	153.6M
compatible	44.1k	32	128M
compatible	44.1k	32	80M
compatible	44.1k	32	26M
compatible	44.1k	16	153.6M

IIS mode	rate	width	clk_iis
compatible	44.1k	16	128M
compatible	44.1k	16	80M
compatible	44.1k	16	26M
compatible	44.1k	8	153.6M
compatible	44.1k	8	128M
compatible	44.1k	8	80M
compatible	44.1k	8	26M
Justified	192k	32	153.6M(Suggest)
Justified	192k	32	128M(Suggest)
Justified	192k	32	80M(Not Suggest)
Justified	192k	32	26M(Not Suggest)
Justified	192k	16	153.6M(Suggest)
Justified	192k	16	128M(Suggest)
Justified	192k	16	80M(Suggest)
Justified	192k	16	26M(Not Suggest)
Justified	192k	8	153.6M
Justified	192k	8	128M
Justified	192k	8	80M
Justified	192k	8	26M
Justified	44.1k	32	153.6M
Justified	44.1k	32	128M
Justified	44.1k	32	80M
Justified	44.1k	32	26M
Justified	44.1k	16	153.6M
Justified	44.1k	16	128M
Justified	44.1k	16	80M
Justified	44.1k	16	26M
Justified	44.1k	8	153.6M
Justified	44.1k	8	128M
Justified	44.1k	8	80M
Justified	44.1k	8	26M

NOTE

In IIS fraction clock divide model1, clk_iis frequency should be higher when bit clock is fast.

8.13.5.4 Working mode examples

1. DMA mode,PCM,Slave,16bit

- iis_clkd = 32'0000_000a
- iis_ctl0 = 32'h0000_c3d9
- iis_ctl1 = 32'h0000_0f11
- iis_ctl2 = 32'h0000_0001
- iis_ctl3 = 32'h0000_0f0f (threshold is half volume of buffer)
- iis_int_en=32'h0000_0036

2. Interrupt mode,PCM,slave,16bit

- iis_clkd = 32'0000_000a
- iis_ctl0 = 32'h0000_83d9
- iis_ctl1 = 32'h0000_0f11
- iis_ctl2 = 32'h0000_0001
- iis_ctl3 = 32'h0000_0f0f (threshold is half volume of buffer)
- iis_int_en=32'h0000_00f6

NOTE

IIS is same configuration as pcm mode.

8.13.5.5 Debug tips

- To ensure that it is working well, the first step is to test iis_sck, iis_lrck, iis_di, iis_do using scope or by observing state of IIS_STS2.
- In the IIS mode, iis_lrck changes periodically, and high/low duty ratio is 50%. In the pcm mode, iis_lrck is a periodical pulse.
- If iis_sck, iis_lrck, iis_di, iis_do is normal, then you can operate IIS_TXD. Transmitting a word by writing data to IIS_TXD in one side(e.g. transmitter). Observe the result by reading data from IIS_TXD in another side(e.g. receiver).
- IIS working on fraction mode1 and channel width = 32bit, IIS_DUMMY_DEBUG is set to be 1. And in other cases, IIS_DUMMY_DEBUG should set to be 0. IIS_DUMMY_DEBUG bit is added.

8.14 I2C

8.14.1 Overview

2 wire serial system is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. It is most suitable for applications requiring occasional communication over a short distance between many devices.

The interface defines 3 transmission speeds:

- Standard mode: 100 kbit/s
- Fast mode: 400 kbit/s

- High-speed mode: 3.4 Mbit/s

8.14.2 Features

I2C features list:

- Software programmable clock frequency
- Interrupt driven data-transfers
- Start/Stop/Repeated Start/Acknowledge generation
- Support Clock Stretching/Wait state generation
- Single Master Operation
- Standard, fast and High-speed modes are supported in this design
- Software programmable duty ratio of clock frequency are supported in this design
- Separate data and command register, Read and write data use the same FIFO
- According to different communication frequency, Start/stop setup and hold, stop hold timing can adjust
- Support DMA operation mode
- Support Hardware channel mode

8.14.3 Function Description

Figure 8-74 I2C system diagram

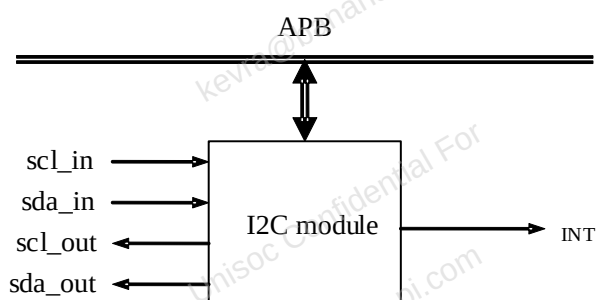


Table 8-20 I2C interrupt

Int classification	Description
I2c_sts[0]	Stop done
	Receive the wrong ack.
	Write done but not stop. (for restart)
Fifo_full_int	FIFO almost full & full_int_en
Fifo_empty_int	FIFO almost empty & empty_int_en

8.14.3.1 System Configuration

The 2 wire serial system uses a serial data line (SDA) and a serial clock line (SCL) for data transfers. All devices connected to these two signals must have open drain or open collector outputs. The logic AND function is exercised on both lines with external pull-up resistors.

The 2 wire serial controller implemented here is a single master device; therefore it starts generating a clock as soon as it is enabled. The user should program this register to the desired value before starting any transfers.

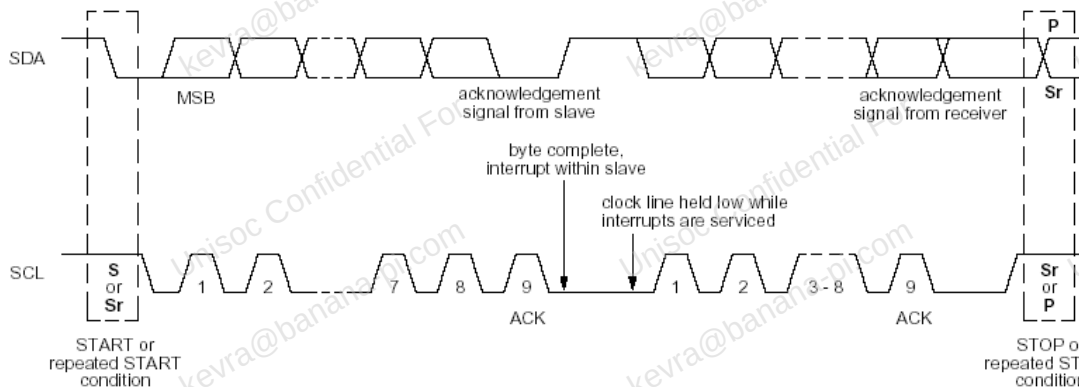
Data is transmitted synchronously to SCL on the SDA line on a Byte-by-Byte basis. Each data Byte is 8 bits long. There is one SCL clock pulse for each data bit with the MSB being transmitted first. There is an acknowledge bit following each transferred Byte. Each bit is sampled during the high period of SCL; therefore the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (START or STOP command).

8.14.3.2 Wire Serial Protocol

Normally, a standard communication consists of four parts:

- START signal generation
- Slave address transfer
- Data transfer
- STOP signal generation

Figure 8-75 Data transfer on the I2C-bus



1. START signal

When the bus is free/idle, that is no master device is engaging the bus (both SCL and SDA lines are high), a master can initiate a transfer by sending a START signal. A START signal is defined as a high-to-low transition of SDA while SCL is high. The START signal denotes the beginning of a new data transfer.

A repeated START is a START signal without first generating a STOP signal. The master uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. writing to device to reading from device) without releasing the bus.

The controller generates a START signal when the start bit in the `i2c_ctl` register is set and the read or write bits are set. Depending on the current status of the SCL line a START or Repeated START is generated.

2. Slave Address Transfer

The first Byte of data transferred by the master immediately after the START signal is the slave address. This is a seven-bit calling address followed by a RW bit. The RW bit signals the slave data transfer direction. No two slaves in the system can have the same address. Only the slave with an address that matches the one

transmitted by the master will respond by returning an acknowledge bit by pulling the SDA low at the 9th SCL clock cycle.

The controller treats a Slave Address Transfer as any other write action. Store the slave device's address in the `i2c_addr_cfg` register. The controller will then transfer the slave address on the bus.

3. Data Transfer

Once successful slave addressing is achieved, the data transfer can proceed on a Byte-by-Byte basis in the direction specified by the RW bit sent by the master. Each transferred Byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a No Acknowledge (NACK), the master can generate a STOP signal to abort the data transfer or generate a repeated START signal and start a new transfer cycle.

If the master, as the receiving device, does not acknowledge the slave, the slave releases the SDA line for the master to generate a STOP or repeated START signal.

For writing data to a slave, store the data to transmit in the FIFO by `i2c_tx` register write. For reading data from a slave, set the read bit. When the transfer is done, an interrupt is generated to MCU. Read data from the `i2c_rx` register.

4. STOP signal

The master can terminate the communication by generating a STOP signal. A STOP signal is defined as a low-to-high transition of SDA while SCL is at logical '1'.

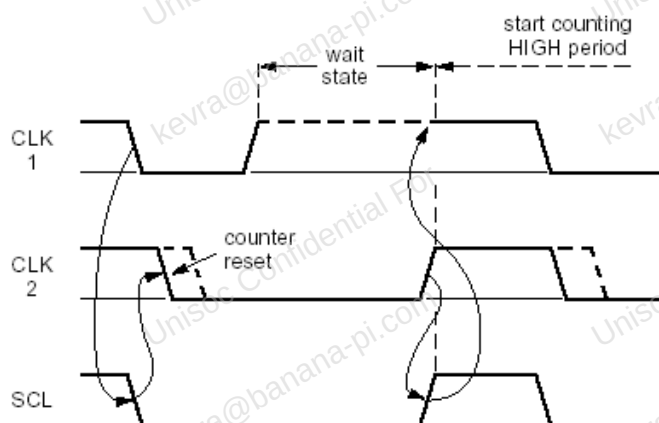
8.14.3.3 Arbitration Procedure

Since the 2 wire serial controller supports single master configurations only, no Arbitration logic is added to the controller. Only clock synchronization is supported since slave devices can use this mechanism for clock stretching.

1. Clock Synchronization

Since the logical AND function is performed on the signals, a high to low transition on SCL or SDA affect all devices connected to the bus. The SCL clock signal can be synchronized between multiple masters using this feature. Each device starts counting its SCL low period when the current master drives SCL low. Once a device's clock has gone low, it holds the SCL line low until the clock high state is reached.

Figure 8-76 Clock synchronization during the arbitration procedure



2. Clock Stretching

Slave devices can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL low the slave can drive SCL low for the required period and then release it. If the slave's SCL low period is greater than the master's SCL low period, the resulting SCL bus signal low period is stretched, thus inserting wait-states.

8.14.3.4 Timing Parameters

Figure 8-77 I2C timing diagram

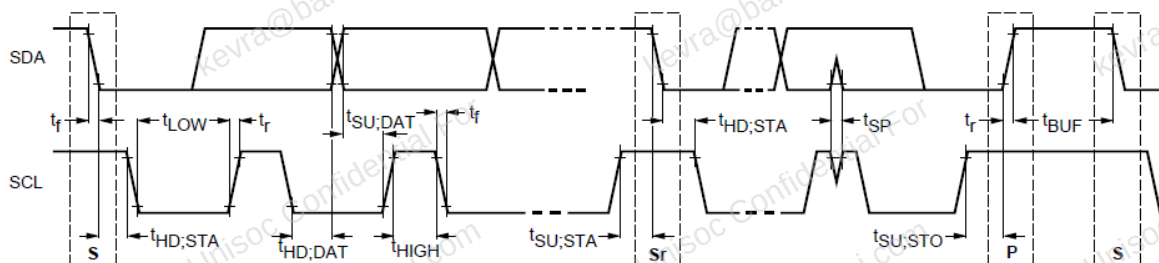


Table 8-21 I2C timing diagram

Parameter	Description	Standard mode		Fast mode	
		Min	Max	Min	Max
fSCL	SCL clock frequency	0kHz	100kHz	0kHz	400kHz
tHD;STA	Hold time (repeated) START condition. After this period, the first clock pulse is generated	4.0μs	-	0.6μs	-
tLOW	LOW period of the SCL clock	4.7μs	-	1.3μs	-
tHIGH	HIGH period of the SCL clock	4.0μs	-	0.6μs	-
tSU;STA	Set-up time for a repeated START condition	4.7μs	-	0.6μs	-
tHD;DAT	Data hold time	0μs	3.45μs	0	0.9μs
tSU;DAT	Data set-up time	250ns	-	100ns	-
tr	Rise time of both SDA and SCL signals	-	1000ns	20 + 0.1Cb	300ns
tf	Fall time of both SDA and SCL signals	-	300ns	20 + 0.1Cb	300ns
tSU;STO	Set-up time for STOP condition	4.0μs	-	0.6μs	-
tBUF	Bus free time between a STOP and START condition	4.7μs	-	1.3μs	-
Cb	Capacitive load for each bus line	-	400pF	-	400pF

8.14.3.5 DMA flow

I2c master support DMA operation:

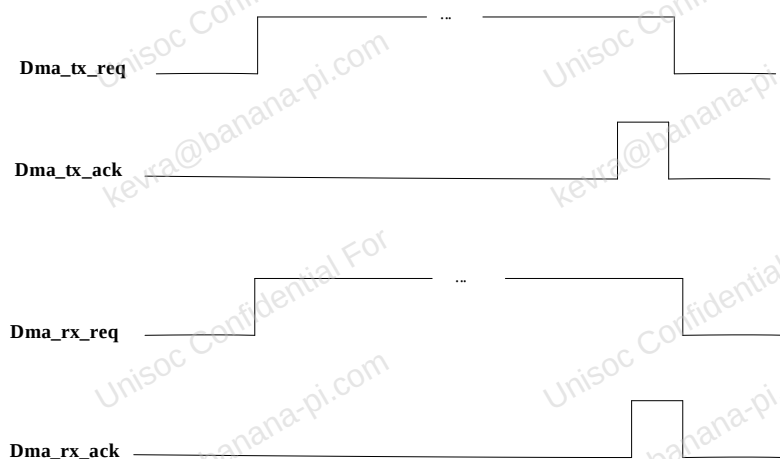
- **DMA SEND**
start DMA transmission, Configure i2c_count register to determine the number of transmission, configure i2c_ctl[15:12] to set tx_fifo empty threshold, once FIFO's empty condition is meet, dma_tx_req will be

valid and note DMA send data to i2c. After dma transfer enough data, dma send the dma_tx_ack signal to end the request of send; At the same time the i2c begin transmitting data. When the FIFO is empty again, request to send again.

- **DMA RECEIVE**

start DMA transmission, Configure i2c_count register to determine the number of receive, configure i2c_ctl[19:16] to set rx_fifo full threshold, once FIFO's full condition is meet, dma_rx_req will be valid and note DMA receive data from i2c. After dma receive enough data, dma send the dma_rx_ack signal to end the request of receive. When the FIFO is full again, request to send again.

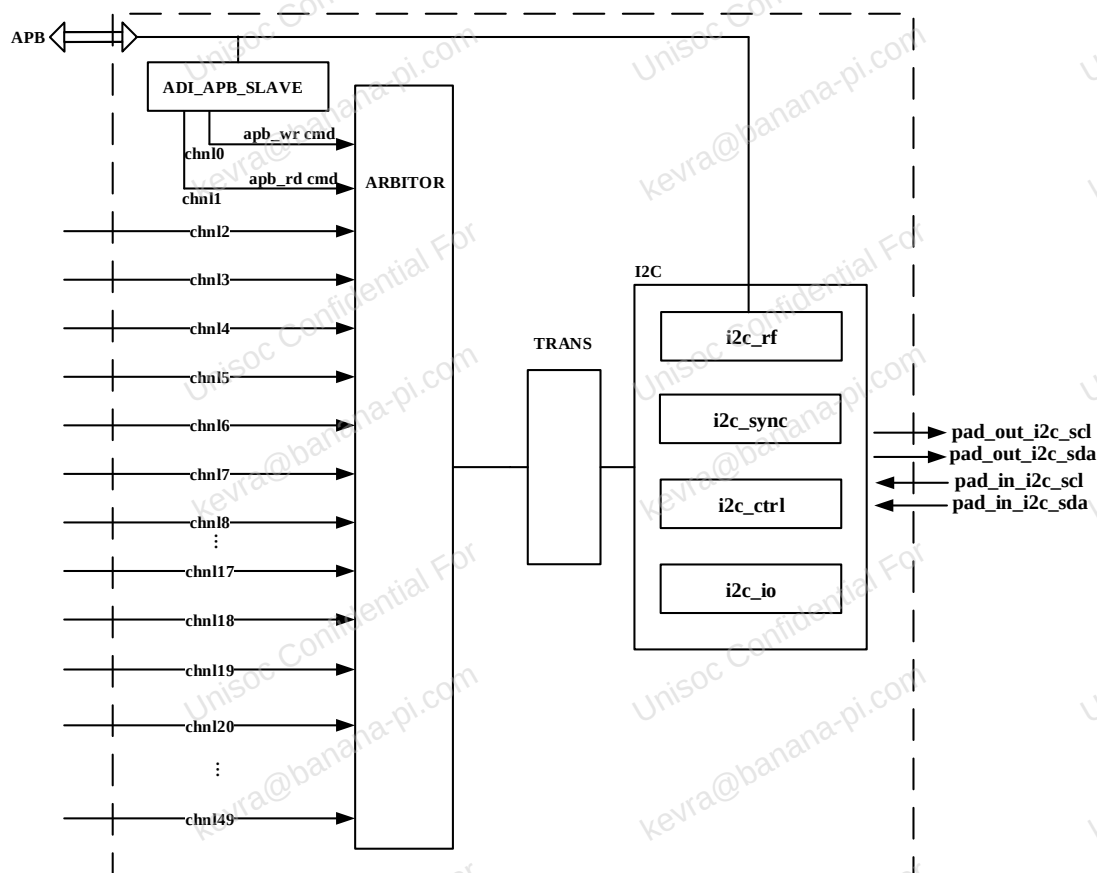
Figure 8-78 DMA transmission mode



8.14.3.6 Hardware channel mode

Hardware channel is supported in this version of i2c, which is a new function that enables the hardware channels to send a tx or rx request or event triggered signal to i2c directly, instead of APB bus.

Figure 8-79 Architecture



Due to the feature of configurable hardware channels, bus signal is used to integrate the hardware channel signals, here the order of signals is provided:

- **hw_tx_req[11:0]** = {hw11_tx_req, hw10_tx_req, hw9_tx_req, hw8_tx_req, hw7_tx_req, hw6_tx_req, hw5_tx_req, hw4_tx_req, hw3_tx_req, hw2_tx_req, hw1_tx_req, hw0_tx_req};
- **hw_tx_data[191:0]** = {hw11_tx_data, hw10_tx_data, hw9_tx_data, hw8_tx_data, hw7_tx_data, hw6_tx_data, hw5_tx_data, hw4_tx_data, hw3_tx_data, hw2_tx_data, hw1_tx_data, hw0_tx_data};
- **hw_tx_ack[11:0]** = {hw11_tx_ack, hw10_tx_ack, hw9_tx_ack, hw8_tx_ack, hw7_tx_ack, hw6_tx_ack, hw5_tx_ack, hw4_tx_ack, hw3_tx_ack, hw2_tx_ack, hw1_tx_ack, hw0_tx_ack};
- **hw_rx_req[11:0]** = {hw11_rx_req, hw10_rx_req, hw9_rx_req, hw8_rx_req, hw7_rx_req, hw6_rx_req, hw5_rx_req, hw4_rx_req, hw3_rx_req, hw2_rx_req, hw1_rx_req, hw0_rx_req};
- **hw_rx_data[191:0]** = {hw11_rx_data, hw10_rx_data, hw9_rx_data, hw8_rx_data, hw7_rx_data, hw6_rx_data, hw5_rx_data, hw4_rx_data, hw3_rx_data, hw2_rx_data, hw1_rx_data, hw0_rx_data};
- **hw_rx_ack[11:0]** = {hw11_rx_ack, hw10_rx_ack, hw9_rx_ack, hw8_rx_ack, hw7_rx_ack, hw6_rx_ack, hw5_rx_ack, hw4_rx_ack, hw3_rx_ack, hw2_rx_ack, hw1_rx_ack, hw0_rx_ack};

HW_TX

- Number: up to 12
- Signal:

- hw_tx_req // input

- hw_tx_data[15:0] // input
- hw_tx_ack // output

- Data Frame:

S	7 bit Slave Address + R/W=0	ACK	Reg Addr	ACK	Data	ACK	P
---	-----------------------------	-----	----------	-----	------	-----	---

- hw_wr_cmd[33:0] = { hw_wr_addr[16:0], wr_bit, hw_wr_data[15:0] }
- 7 bit slave address = hw_wr_cmd[31:25] = hw_wr_addr[14:8]
- Reg ddr = hw_wr_cmd[24:17] = hw_wr_addr[7:0]
- Data = hw_wr_cmd[7:0]
- Hw_tx_data[15:0] = hw_wr_data[7:0]

NOTE

Totally, in this mode each address contains slave address and register address, the second one is regarded as a data frame in order to adjust to I2C protocol. And each address needs to configure before a hardware channel I2C transmission. Besides, the lower 8 bit in hw_tx_data[15:0] is for Data.

HW_RX

- Number: up to 12
- Signal:
 - hw_rx_req // input
 - hw_rx_data[15:0] // output
 - hw_rx_ack // output
- Data Frame:

S	7 bit Slave Address + R/W=0	ACK	Reg Addr	ACK	...
RS	7 bit Slave Address + R/W=1	ACK	Data	ACK	P

- hw_rd_cmd[33:0] = { hw_rd_addr[16:0], rd_bit, 16'b0 }
- 7 bit slave address = hw_rd_cmd[31:25] = hw_rd_addr[14:8]
- Reg Addr = hw_rd_cmd[24:17] = hw_rd_addr[7:0]
- Data = hw_rx_data[7:0]
- Hw_rx_data[15:0] = hw_rx_data[7:0]

NOTE

Similarly, in hardware channel I2C read need to send slave address and then write register address. Followed by a repeated start, it transmits slave address again and finally read data. Besides, the lower 8 bit in hw_rx_data[15:0] is for Data.

evt_sig_fifo

- Number: up to 12
- Signal: evt_sig
- FIFO depth: 4

- Data Frame:

S	7 bit Slave Address + R/W=0	ACK	Reg Addr	ACK	Data	ACK	P
---	-----------------------------	-----	----------	-----	------	-----	---

- evt_wr_cmd[33:0] = { evt_wr_addr, wr_bit, 15'h0, evt_data };
- 7 bit slave address = evt_wr_cmd[31:25] = evt_wr_addr[14:8]
- Reg ddr = evt_wr_cmd[24:17] = evt_wr_addr[7:0]
- Data = {7'h0, evt_data}

NOTE

- evt_sig could trigger a specific event and the value of evt_data could be 1 or 0 depends on the transition edge. 0->1 means 1, and 1->0 is for 0.

evt_sig_nofifo

- Number: up to 12
- Signal: evt_sig
- evt_wr_cmd[33:0] = { evt_wr_addr, wr_bit, 15'h0, evt_data };
- 7 bit slave address = evt_wr_cmd[31:25] = evt_wr_addr[14:8]
- Reg ddr = evt_wr_cmd[24:17] = evt_wr_addr[7:0]
- Data = {7'h0, evt_data}

ARM_WR_CMD

- Number: 1
- Signal: APB signals
- FIFO depth: 8
- fifo_wdata[33:0] = { arm_wr_addr, 1'b0, wr_data[15:0] }
- arm_wr_addr = arm_addr[16:0] = PADDRadi[17:0]
- wr_data[15:0] = PWDATAadi[15:0]

NOTE

In this mode, ARM_WR_CMD should configure like HW_TX and will be under an arbitration process with other hardware channels. Here arm_wr_addr means the address of register file.

ARM_RD_CMD

- Number: 1
- Signal: APB signals
- Configure 0x013C ARM_RD_CMD, and then read the value from 0x0140 ARM_RD_DATA.

Table 8-22 Channel name and number

Hardware channel	Channel number
ARM_WR_CMD	chnl0
ARM_RD_CMD	chnl1
evt0_fifo	chnl2

Hardware channel	Channel number
evt1_fifo	chnl3
evt2_nofifo	chnl4
evt3_fifo	chnl29
evt4_fifo	chnl30
evt5_fifo	chnl31
evt6_fifo	chnl32
evt7_fifo	chnl33
evt8_fifo	chnl34
evt9_fifo	chnl35
evt10_fifo	chnl36
evt11_fifo	chnl37
evt12_fifo	chnl38
evt13_nofifo	chnl39
evt14_nofifo	chnl40
evt15_nofifo	chnl41
evt16_nofifo	chnl42
evt17_nofifo	chnl43
evt18_nofifo	chnl44
evt19_nofifo	chnl45
evt20_nofifo	chnl46
evt21_nofifo	chnl47
evt22_nofifo	chnl48
evt23_nofifo	chnl49
hw_tx0	chnl5
hw_tx1	chnl7
hw_tx2	chnl9
hw_tx3	chnl11
hw_tx4	chnl13
hw_tx5	chnl15
hw_tx6	chnl17
hw_tx7	chnl19

Hardware channel	Channel number
hw_tx8	chnl21
hw_tx9	chnl23
hw_tx10	chnl25
hw_tx11	chnl27
hw_rx0	chnl6
hw_rx1	chnl8
hw_rx2	chnl10
hw_rx3	chnl12
hw_rx4	chnl14
hw_rx5	chnl16
hw_rx6	chnl18
hw_rx7	chnl20
hw_rx8	chnl22
hw_rx9	chnl24
hw_rx10	chnl26
hw_rx11	chnl28

8.14.4 Control Registers

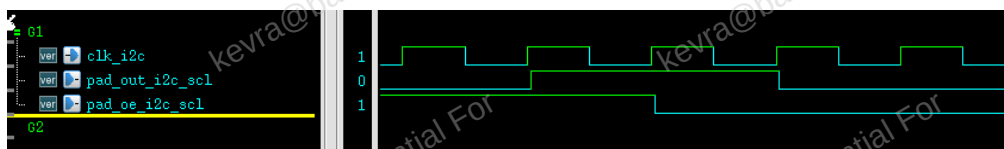
See *UIS7885 AP System Control Registers Instructions*.

8.14.5 Application Notes

8.14.5.1 Skew request check

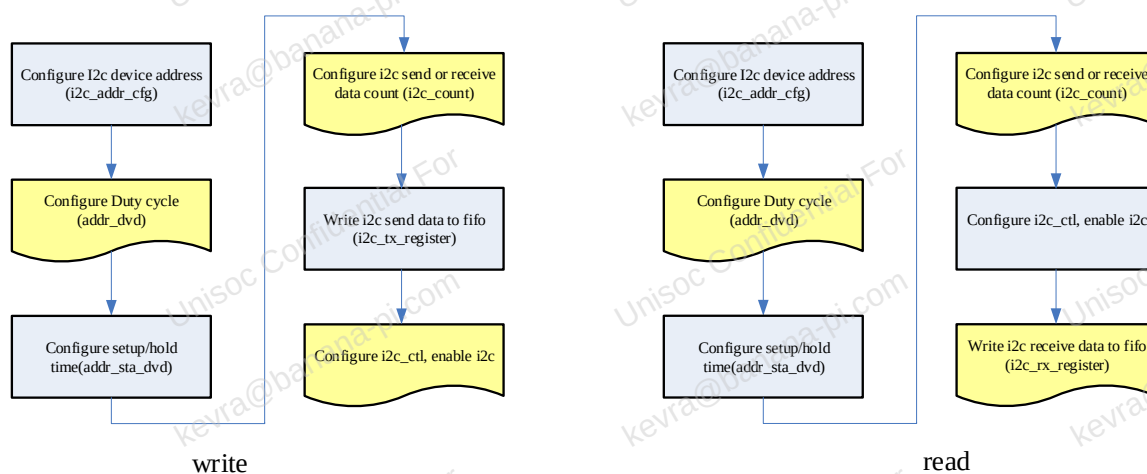
In this design, skew request is necessarily required to be checked, which is also need to take an attention in the following project developments.

For example, the skew request between pad_out_i2c_scl and pad_oe_i2c_scl is period of the fastest I2c clock used in project (e.g. 6ns for 153.6MHz). This can constrain the balance of different paths the outputs go through, so does pad_out_i2c_sda and pad_oe_i2c_sda.



8.14.5.2 Programming Model

Figure 8-80 software control flow



8.14.5.3 100K mode program examples

I2c_clk: 26MHz

1. Configure i2c device address by i2c_addr_cfg;
2. Configure i2c low/high phase and setup/hold time by addr_dvd and addr_sta_dvd;
3. Configure i2c send or receive data count;
4. write i2c send dmatx to FIFO for i2c data sending operation by write i2c_tx register;
5. i2c_ctl to read or write by configure i2c_ctl[3] , enable i2c and other options by i2c_ctl register, i2c transmit start;

For example, configure i2c_addr_cfg to 0x80 and i2c_count is 0x10, start 100k i2c write operation:

1. i2c_addr_cfg = 0x80;
2. i2c_count = 0x10;
3. Write data to i2c_tx register;
4. i2c_addr_dvd0 = 0x003D003D;
5. i2c_ctl = 0x00088007;

For example, configure i2c_addr_cfg to 0x80 and i2c_count is 0x10, start 100k i2c read operation:

1. i2c_addr_cfg = 0x80;
2. i2c_count = 0x10;
3. Read data to i2c_tx register;
4. i2c_addr_dvd0 = 0x003D003D;
5. i2c_ctl = 0x0008800f;

8.14.5.4 400K mode program examples

I2c_clk: 48MHz

1. Configure i2c device address by i2c_addr_cfg;
2. Configure i2c low/high phase and setup/hold time by addr_dvd and addr_sta_dvd;
3. Configure i2c send or receive data count;
4. Write i2c send data to FIFO for i2c data sending operation by write i2c_tx register;
5. i2c_ctl to read or write by configure i2c_ctl[3] , enable i2c and other options by i2c_ctl register, start i2c transmit;

For example, configure i2c_addr_cfg to 0x80 and i2c_count is 0x10 ,start 400k i2c write operation:

1. i2c_addr_cfg =0x80;
2. i2c_count =0x10;
3. Write data to i2c_tx register;
4. i2c_addr_dvd0 =0x00100028;
5. i2c_ctl =0x00088007;

8.14.5.5 HS mode program examples

I2c_clk :153.6M

1. Configure i2c device address by i2c_addr_cfg;
2. Configure i2c low/high phase and setup/hold time by addr_dvd and addr_sta_dvd;
3. Configure i2c send or receive data count;
4. Use i2c high speed mode, need configure i2c_HSMODE_CFG;
5. Write i2c send data to FIFO for i2c data sending operation by write i2c_tx register;i2c_ctl to read or write by configure i2c_ctl[3] , enable i2c and other options by i2c_ctlregister , i2c transmit start;

For example, configure i2c_addr_cfg to 0x80 and i2c_count is 0x10 ,start HS i2c writer operation:

1. i2c_addr_cfg =0x80;
2. i2c_count =0x10;
3. Write data to i2c_tx register;
4. i2c_hsmode_cfg=0x000a0209;
5. i2c_addr_dvd0 =0x00400078;
6. i2c_ctl =0x000d8417;

NOTE

Change the value of prescale register no need clock duty ratio modify .Due to the structure of the I2C interface, the controller uses “4×SCL” clock internally.

Example1:

- PCLK = 26MHz, desired SCL = 100kHz, I2c_dvd_opt=0
- Low_level = (2 × i2c_low_dvd + 8)/26M;
- High_level = (2 × i2c_high_dvd + 8)/26M;
- Low_level > Tlow(min);
- High_level > Thigh (min);
- Low_level + High_level =1/100K;
- i2c_low_dvd =0x3D; i2c_high_dvd =0x3D;

Example2:

- PCLK = 153.6MHz, desired SCL = 3.4M,
- i2c_low_dvd = 0x10; i2c_high_dvd = 0x04;
- PCLK = 26MHz, desired SCL = 100kHz, I2c_dvd_opt=1;
- Low_level = $(2 \times \text{i2c_low_dvd} + 2)/26\text{M}$;
- High_level = $(2 \times \text{i2c_high_dvd} + 2)/26\text{M}$;
- Low_level > Tlow (min);
- High_level > Thigh (min);
- Low_level + High_level = 1/100K;
- i2c_low_dvd = 0x3B; i2c_high_dvd = 0x3B;

8.14.5.6 DMA mode program examples

1. Configure i2c device address by i2c_addr_cfg;
2. Configure i2c low/high phase and setup/hold time by addr_dvd and addr_sta_dvd;
3. Configure i2c send or receive data count;
4. if use i2c high speed mode, need configure i2c_HSMODE_CFG;
5. write i2c send data to FIFO for i2c data sending operation by write i2c_tx register;
6. i2c_ctl to read or write by configure i2c_ctl[3], enable i2c and other options by i2c_ctl register, i2c transmit start;

For example, configure i2c_addr_cfg to 0x80 and i2c_count is 0x10, start DMA i2c writer operation:

1. i2c_addr_cfg = 0x80;
2. i2c_count = 0x10;
3. DMA write data to i2c_tx register;
4. i2c_addr_dvd0 = 0x003D003D;
5. i2c_ctl = 0x00088707;

8.14.5.7 Hardware channel mode program examples

1. Configure i2c low/high phase and setup/hold time by addr_dvd and addr_sta_dvd;
If use i2c high speed mode, need to configure i2c_HSMODE_CFG;
2. Configure HW_CTL, 32'h30 or 32'h10, according to cmd_wr_addr_en.
3. configure CHNL_EN0 and CHNL_EN1;
4. HW_CHNL_PRI registers, if necessary;
5. Configure hardware channel input, according to the kinds of channels (HW_TX, HW_RX, evt_sig_fifo, evt_sig_nofifo);

For example: **hw0_tx** works in i2c standard mode (channel 5)

1. After reset, addr_dvd and addr_sta_dvd will be reset to i2c standard mode.
2. Set HW_CTL to 32'h16 (for 15 bit address mode)
3. Configure CHNL_EN0 = 32'b100000 (enable the transfer of channel 5).
4. HW_CHNL_PRI could be configured if necessary. (e.g. HW_CHNL_PRI[17:15] = 3'h4)
5. Set hw_tx_req, hw_tx_data[15:0] and wait hw_tx_ack to pull down hw_tx_req

For example: **hw0_rx** works in i2c standard mode (channel 6)

1. After reset, addr_dvd and addr_sta_dvd will be reset to i2c standard mode.

2. Set HW_CTL to 32'h20.
3. Configure CHNL_EN0 = 32'h1000000.
4. HW_CHNL_PRI could be configured if necessary. (e.g. HW_CHNL_PRIL[20:18] = 3'h4)
5. Set hw_rx_req, and wait hw_tx_ack to get hw_rx_data.

8.14.5.8 Hardware channel mode program examples (APB_WR_CMD, APB_RD_CMD)

1. Configure i2c low/high phase and setup/hold time by addr_dvd and addr_sta_dvd;
2. If use i2c high speed mode, need to configure i2c_HSMODE_CFG;
3. Configure HW_CTL, 32'h48 or 32'h16, according to cmd_wr_addr_en.
4. Configure HW_CHNL_PRI registers, if necessary;
5. Configure APB command channels. For write, set ARM_CMD_WR's chnl_addr0 and AMR_DAT_WR's chnl0_data if cmd_wr_addr_en is 1'h1 and using command mode. If cmd_wr_addr_en is 0, PADDR and PWDATA will be regarded as address and data that transmits on the bus.
6. For read, set PADDR[8:0] = 0x013C, then it triggers a APB read request.
Data can be read from ARM_RD_DATA after arm_rd_cmd_busy is 0.

8.14.5.9 Programming Notes

- When change I2C control register configuration, SW should ensure:
 - a tx_fifo is real empty
 - b rx_fifo is real empty
 - c Change I2C control register
- DMA enable
 - a DMA should be configured
 - b Program I2C control register
 - c Write I2C_ctl[0]

8.15 UFS

8.15.1 Overview

The Cores UFS Host Controller IP is a high-performance interface that is primarily used in mobile devices where data is stored on non-volatile mass storage memory devices.

8.15.2 Features

- Scatter/Gather DMA to transfer large data blocks
- Burst transfers to maximize DMA throughput and keep system impact minimal
- Pre-configured for up to 32 task requests
- Pre-configured for up to 8 task management requests
- Ability to perform commands without system host intervention
- Support to pre-fetch more than one PRD entry (up to 16 PRD entries)
- Support for the full range of UPIU packets, from 32 Byte up to 64 kB

- 32- APB4 slave interface for system host programming access:
- 128 AXI master interface for DMA controlled data transfer:
- Registered interface with a minimal amount of logic between port and register
- Hierarchical architecture to enable the design for customer owned technology dependent M-PHY
- Support the following UFS2.0 features
 - Higher speed up to HS-G3 (High-Speed Gear 3)
 - Symmetric 2RX-2TX lanes
 - Pre-configured for up to 2 TX/RX lanes
- Auto-hibernate entry and exit sequence
- Support the following UFS2.1 features.
 - Unified Memory Extension (UME)
 - Auto Hibernate flow in UM operation
 - Inline Encryption with AES-XTS algorithm only
- Support the following UFS 3.0 features
 - Higher speed up to HS-G4 (High Speed Gear 4)
 - Cryptography engine for HS G4 throughput
 - PA_INIT error code support in UECDL register
 - Support for QoS indication for TX, RX and PA_INIT in UECDME register
- AXI Master Outstanding Writes
- Multiple Outstanding RTTs up to 16.
- Support all the UniPro 1.61 and M-PHY 3.1 features including:
 - SKIP symbol insertion
 - MK2 extension support
 - Scrambling for EMI mitigation
 - HS-Gear3 adaption
 - Advanced granularity support
- Support all UNIPRO 1.8 and MPHY 4.1 features (These features require DWC-UFSHC-V30 and DWC-UFSHC-UNIPRO-V18 licenses) including:
 - HS G4 Support
 - QoS Indication
 - New error Code PA_INIT defined for Data Link Layer
 - Revised Link Startup Timeout conditions
 - Deprecated support for Optical Media Converters being programmed in line through MPHY Line-Config state
- Support HS Gear4 Adapt functionality
- Support Fatal Error Interrupt signal, which reports only the fatal error events
- Support multiple AXI RID for Master Reads
- Support WDP prefetch
- Support UFS Card Detection (This feature requires DWC-UFSHC-V21 or DWC-UFSHC-V30 license)

Unsupported Features: Out of order data delivery for UTP data transfer.

9 Audio Subsystem

9.1 General Description

9.1.1 Overview

Audio sys is primarily used for audio processing, assisting AP sys, reducing the burden on the AP CPU. AUD DSP is responsible for encoding and decoding, it can handle calls, audio playback, etc, almost all voice are processed by audio sys.

9.1.2 Audio Path Features

The device is a flexible, low-power, low-voltage, high quality stereo audio codec, which containing a mono Analog to Digital Converter (ADC), a stereo Digital to Analog Converter (DAC) and additional analog circuitry (Note: Codec Analog Part is not implemented in this chip). The feature of the digital part includes:

- The device can cover operations from 8 kHz mono voice playback to 48kHz stereo audio playback
- The record path covers operations from 8 kHz to 48 kHz stereo recording
- The playback path offers signal-processing blocks for filtering and effects, and supports flexible mixing of DAC signals
- Looping path is supported for various applications such as FM, Bluetooth and ATE test
- Low power playback and recording is supported by offloading to the DSP core
- Extensive Register based control of power, input/output channel configuration, gains, effects, pin-multiplexing, and clocks is included, allowing the device to be precisely targeted to its application.
- Programmable sampling frequency F_s for ADC: 8/12/16/20/24/28/32/36/40/44/48/52 kHz
- Programmable sampling frequency F_s for DAC: 8/9.6/11.025/12/16/22.05/24/32/44.1/48/96 kHz
- External CODEC is also supported by connecting through the IIS interface
- DMIC is supported through PDM interface
- External smart PA is supported by connecting with VBC IIS master

The voice band control (VBC) interface communicates with software or DMA, and this block contains data buffers for each channel, audio signal processing blocks, APB bus for AP or CP access, interrupt control and DMA interface control logic.

The audio codec digital part is responsible for signal up-sample interpolation(DAC) and down-sample decimation(ADC). Also PDM interface is designed for DMIC application.

Several groups of IIS serial interface connect ADC/DAC of the audio codec with the VBC interface, two groups are for ADC channels, and the other is for DAC channels. The VBC interface is always the slave, while the audio codec part is the master.

Except the VBC interface, all the other modules together constitute the audio subsystem.

- The SRC_48K module is for DAC signal sampling rate converted to 48K
- The MCDT module is for transferring audio data with other subsystems(AP and communication)
- PDM is for DMIC Voice Trigger and DMIC Context Hub

- Two DMA is implemented. DMA_AP is for AP CPU, which is mainly used in MP3 playback. DMA_CP is for audio DSP core
- IIS module is implemented for audio data transmission with external chips
- UART module is for DSP debugging and Bluetooth A2DP applications
- Spinlock is for security of multi-core accessing VBC
- Watchdog is for abnormal system reset
- VAD is for AMIC Voice Trigger and AMIC Context Hub
- TDM is for Array MIC

9.1.3 System Addr mapping

Audio sys address range: 0x5600_0000~0x57FF_FFFF, total: 32MB. Address is assigned as follows, all address are global.(all device can be accessed by other system through aon)

Table 9-1 Audio sys address

Module Name	Description	Space Allocation	Access System(IP Address)	
			Start Address	End Address
GLB REG	S2 (APB)	64KB	0x5620_0000	0x5620_FFFF
CLOCK TOP REG		64KB	0x5624_0000	0x5624_FFFF
DVFS REG		16KB	0x5625_0000	0x5625_3FFF
Matrix CFG		4KB	0x5625_4000	0x5625_4FFF
Reserved			0x5625_6000	0x5626_FFFF
UART		128Byte	0x5627_0000	0x5627_007F
Reserved			0x5627_0400	0x5627_07FF
IIS0		128Byte	0x5627_0800	0x5627_087F
IIS1		128Byte	0x5627_0900	0x5627_097F
IIS2		128Byte	0x5627_0A00	0x5627_0A7F
ADSP Timer0		32Byte	0x5627_0C00	0x5627_0C1F
ADSP Timer1		32Byte	0x5627_0C80	0x5627_0C9F
ADSP Watch Dog		64Byte	0x5627_1000	0x5627_103F
ICU		128Byte	0x5627_2800	0x5627_287F
reserved			0x5627_2C00	0x562F_FFFF
VBC_s2		256KB	0x5630_0000	0x5633_FFFF
MCDDT_s2		4KB	0x5634_0000	0x5634_0FFF
SRC48K		4KB	0x5635_0000	0x5635_0FFF
AUD		2KB	0x5636_0000	0x5636_07FF
PDM		64KB	0x5637_0000	0x5637_FFFF

Module Name	Description	Space Allocation	Access System(IP Address)	
			Start Address	End Address
VAD		64KB	0x5638_0000	0x5638_FFFF
AON APBREG		16KB	0x5639_0000	0x5639_3FFF
AON CLK TOP REG		1KB	0x5639_4000	0x5639_43FF
TDM		2KB	0x563A_0000	0x563A_07FF
reserved			0x563B_2000	0x563F_FFFF
reserved	S3 (APB)		0x5640_0000	0x564F_FFFF
MCDT_s3		4KB	0x5650_0000	0x5650_0FFF
VBC_s3		1KB	0x5651_0000	0x5651_03FF
reserved			0x5652_0000	0x565F_FFFF
DMA_AP slave	S4 (AHB)	16KB	0x5665_0000	0x5665_3FFF
DMA_CP slave		16KB	0x5666_0000	0x5666_3FFF
reserved			0x56670000	0x5667_FFFF
Spinlock		4KB	0x5668_0000	0x5668_0FFF
reserved			0x5669_0000	0x566F_FFFF
DRAM	S5 (AXI)	384KB	0x5600_0000	0x5605_FFFF
PRAM		128KB	0x5606_0000	0x5607_FFFF
AUD_IRAM0	S6 (AXI)	80KB	0x5680_0000	0x5681_3FFF
AUD_IRAM1(Ret)		32KB	0x5682_0000	0x5682_7FFF
reserved		96KB	0x5682_8000	0x5683_FFFF
reserved			0x5684_0000	0x57FF_FFFF
AON Space	S1 to AON	128MB	0x6000_0000	0x77FF_FFFF

NOTE

1KB = 1024Byte; 1MB = 1024×1024Byte.

9.1.4 System Interrupt

There are some interrupts on audio sys.

Table 9-2 Interrupts of the audio sys

INT source name	I/O	IRQ	IRQ Type	FIQ	FIQ Type	AP_CPU INTSerialNum	PSCP_CPU INTSerialNum	SP_CPU INTSerialNum	CH_CPU INTSerialNum	PHYCPU INTSerialNum	AUD_DSP INTSerialNum
int_req_aud_wdg_rst_extd	O	Y	H	Y	H	60		41			
int_req_vbc_audply	O	Y	H	Y	H	59		40	36		
int_req_vbc_audrcd	O	Y	H	Y	H	58		39	35		
int_req_mcdt_ap	O	Y	H	Y	H	57		38	34		
int_req_dmaap	O	Y	H	Y	H	56		37	33		26
dma_chn2_grp2_int	O	Y	H	Y	H	55		36			
dma_chn1_grp2_int	O	Y	H	Y	H	54		35			
dma_chn2_grp1_int	O	Y	H	Y	H	53		34			
dma_chn1_grp1_int	O	Y	H	Y	H	52		33			
int_req_tdm_tx	O	Y	H	Y	H	51					7
int_req_tdm_rx	O	Y	H	Y	H	50					6
eic_int_ap	O	Y	H	Y	H	61					
int_req_pdm_rx	O	Y	H	Y	H				31		9
int_req_aud_mcdt_pscp1	O	Y	H	Y	H		41				
int_req_aud_mcdt_phycp0	O	Y	H	Y	H					191	
int_req_aud_mcdt_pscp0	O	Y	H	Y	H		42				
int_req_aud_iis2		Y	H								46
int_req_aud_iis1		Y	H								45
int_req_aud_iis0		Y	H								44
int_req_aud_vbc_dac1		Y	H								43

INT source name	I/O	IRQ	IRQ Trig Type	FIQ	FIQ Trig Type	AP_CPU INTSerialNum	PSCP_CPU INTSerialNum	SP_CPU INTSerialNum	CH_CPU INTSerialNum	PHYCPU INTSerialNum	AUD_DSP INTSerialNum
int_req_aud_vbc_dac0		Y	H								42
int_req_aud_vbc_adc3		Y	H								41
int_req_aud_vbc_adc2		Y	H								40
int_req_aud_vbc_adc1		Y	H								39
int_req_aud_vbc_adc0		Y	H								38
int_req_aud_vbc_aud1		Y	H								37
int_req_aud_vbc_aud0		Y	H								36
int_req_aud_vbc	O	Y	H	Y	H				32		35
int_req_aud_mcdt_int1_0		Y	H								32
int_req_aud_dsp_src_irq		Y	H								31
int_req_aud_uart		Y	H								30
int_req_aud_wdg		Y	H								27
int_req_aud_ceva_x_timer		Y	H								25
int_req_aud_tl42x_dmss_dma		Y	H								24
int_req_aud_tl42x_pmss_dma		Y	H								23
int_req_aud_dma_chn1		Y	H								22
dsp_l2cc_int_or		Y	H								21
int_req_aud_vbc_dac1_underflow		Y	H								18
int_req_aud_usb_crc_err		Y	H								17

INT source name	I/O	IRQ	IRQ Trig Type	FIQ	FIQ Trig Type	AP_CPU INTSerialNum	PSCP_CPU INTSerialNum	SP_CPU INTSerialNum	CH_CPU INTSerialNum	PHYCPU INTSerialNum	AUD_DSP INTSerialNum
int_req_aud_dma_chn2		Y	H								15
int_req_aud_busmon	O	Y	H					60			12
int_req_aud_aud_4ad		Y	H								11
int_req_aud_dma_cp		Y	H								10
int_req_vad_flag		Y	H								8
vad_int_full		Y	H								5
vad_int_empty		Y	H								4
vad_int_pulse		Y	H								
int_req_top_dvfs	I	Y	H	Y	H	227	44			215	3
int_req_mailbox_inout5_aud	I	Y	H	Y	H						14

9.1.5 System Power Design

There are two shutdown domains and one AON domain inside the AUDIO system. The shutdown power domains are powered by VDDMM1, and the AON domain was powered by VDDCORE. RAMS in all power domains in AUDCP system are dual rail rams , the SRAM are powered by VDDSRAM.

All the power domains has internal power switches.

- Power domain ON sequence: PD_AUDCP_SYS -> PD_AUD_CEVA
- Power domain OFF sequence: PD_AUD_CEVA -> PD_AUDCP_SYS

9.1.5.1 Power Scheme

Figure 9-1 Power Arch

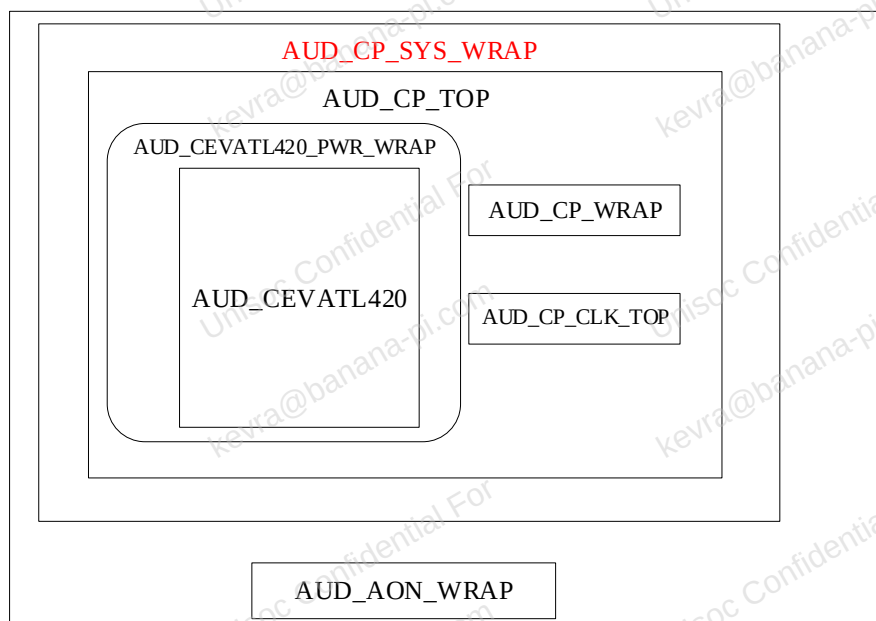


Table 9-3 DVFS Range

Power Rail	Domain	DVFS Range
VDDCORE	AUD_AON	0.75V/0.65V(pmu)
VDDMM1	AUD_CP AUD_CEVA	0.75V/0.70V/0.65V
VDDSRAM	IRAM_ret/DSP_PRAM/DSP_DRAM IRAM/L1CACHE/L2CACHE	No dvfs

9.1.5.2 Power State

Table 9-4 Power states in AUDCP SYS

POWER MODE	PD_AUDCP_SYS	AUDCP RAM_normal	AUDCP RAM_Ret	PD_AUD_CEVA	AUD_CEVA RAM_normal	AUD_CEVA RAM_ret	AO N
ALL_OFF	OFF(PG)	OFF(PG)	ON	OFF(PG)	OFF(PG)	ON	ON
AUDCP_SYS ON	ON (PG)	ON (PG)	ON	OFF(PG)	OFF (PG)	ON	ON
ALL ON	ON	ON	ON	ON	ON	ON	ON

9.2 Processor

9.2.1 Overview

Ceva Teaklite420 DSP Core is used as the audio processor.

9.2.2 Features

The DSP Core clock frequency is 768.8M at most, and 128KB L2Cache is implemented for the core. To support voice trigger, DSP's TCM is increased. PRAM is 128KB, DRAM is 384KB, ICache RAM is 32KB, DCache RAM is 32KB.

Figure 9-2 CEVA-TL420 MEM

PRAM 128KB(Ret)	DRAM 384KB(Ret)	SLV
768MHz	AUD TL420 L2Cache 128KB	
ICache 32KB	DCache 32KB	IO

9.3 Voice Trigger

9.3.1 Overview

In order to achieve Voice Trigger, voice recognition has been added. This algorithm is implemented on AUDIO DSP. In quiet mode, DSP is in power down mode, helping the chip save power. It owns two Voice Trigger features: Digital Voice Trigger and Analog Voice Trigger. On the basis of the original deep sleep current in standby, the power consumption target is to turn on Voice Trigger, with an average increase of no more than 1.5mA.

9.3.2 Features

- Support XTL26M, RCO6/8M main clock Frequency for PDM, XTL2M, RCO2M main clock Frequency for VAD
- Support 1DMIC interfaces for DMIC Voice Trigger
- Support 2channel for AMIC Voice Trigger
- Support 3DMIC interfaces for DMIC record

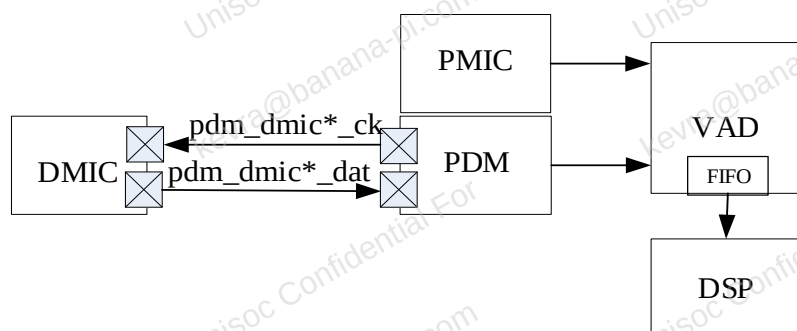
9.3.3 Function Description

Voice Trigger own DMIC Voice Trigger and AMIC Voice Trigger.

- DMIC Voice Trigger includes DMIC, PDM, VAD, DSP, AMIC Voice Trigger includes PMIC, VAD, DSP. Digital Voice Trigger is primarily used.

- The DMIC receives an external digital data, The PDM is the digital part for processing the input of DMIC, its data is send to VAD. VAD is implemented into 2 parts: 1st step and 2nd step. The blocks of 1st step finish the pre-process of voice signal, and compare the energy of the signal with the threshold. The VAD will give a decision that the input signal is human voice or not in 2nd step. Wakeup signal will be generated to CPU or DSP core, to indicate that human voice is detected. The data buffer in 2nd step will buffer the voice data for CPU or DSP core. DSP reads data from the data buffer, then start voice algorithm recognition. If the recognition is successful, audio sys will wake up AP CPU.
- PMIC receives analog data and convert it digital data, finally its data is send to vad too.

Figure 9-3 DMIC and AMIC Data Flow



9.4 Audio codec digital parts

9.4.1 Overview

The aud_4ad_top is the digital part of audio codec. The 2 DAC paths and 4 ADC paths are implemented in aud_4ad_top module. The audifd_top module is also included in aud_4ad_top module. The audifd_top module transfers sigma-delta(SDM) and DNS output data into serial data.

9.4.2 Features

- Support 26MHz main clock Frequency
- Support programmable Fs for DAC: 96/48/44.1/32/24/22.05/16/12/11.025/9.6/8 kHz
- Support programmable Fs for ADC: 8/12/16/20/24/28/32/36/40/44/48/52 kHz
- Support 5 IIS master interfaces for DAC and ADC channel:
 - dac iis0, dac iis1, dac iis2, dac iis3, dac iis4
 - adc0 iis0, adc0 iis1, adc0 iis2, adc0 iis3, adc iis4
 - adc1 iis0, adc1 iis1, adc1 iis2, adc1 iis3, adc iis4
- Support both left and right DAC channel for stereo play, left and right ADC channel for stereo MIC
- Support mute control for DAC channel
- Support DNS(Dynamic Noise Suppression) function for left and right DAC channel
- Support Audio loop path for Production

- Support DAC loop to ADC in Digital part for debug

9.4.3 Function Description

9.4.3.1 AUD Digital Data Path

4 ADC path input data can be selected from aud_ifd, and the output data interface is IIS. 2 DAC path input data are from IIS interface, and the output data are sent to aud_ifd.

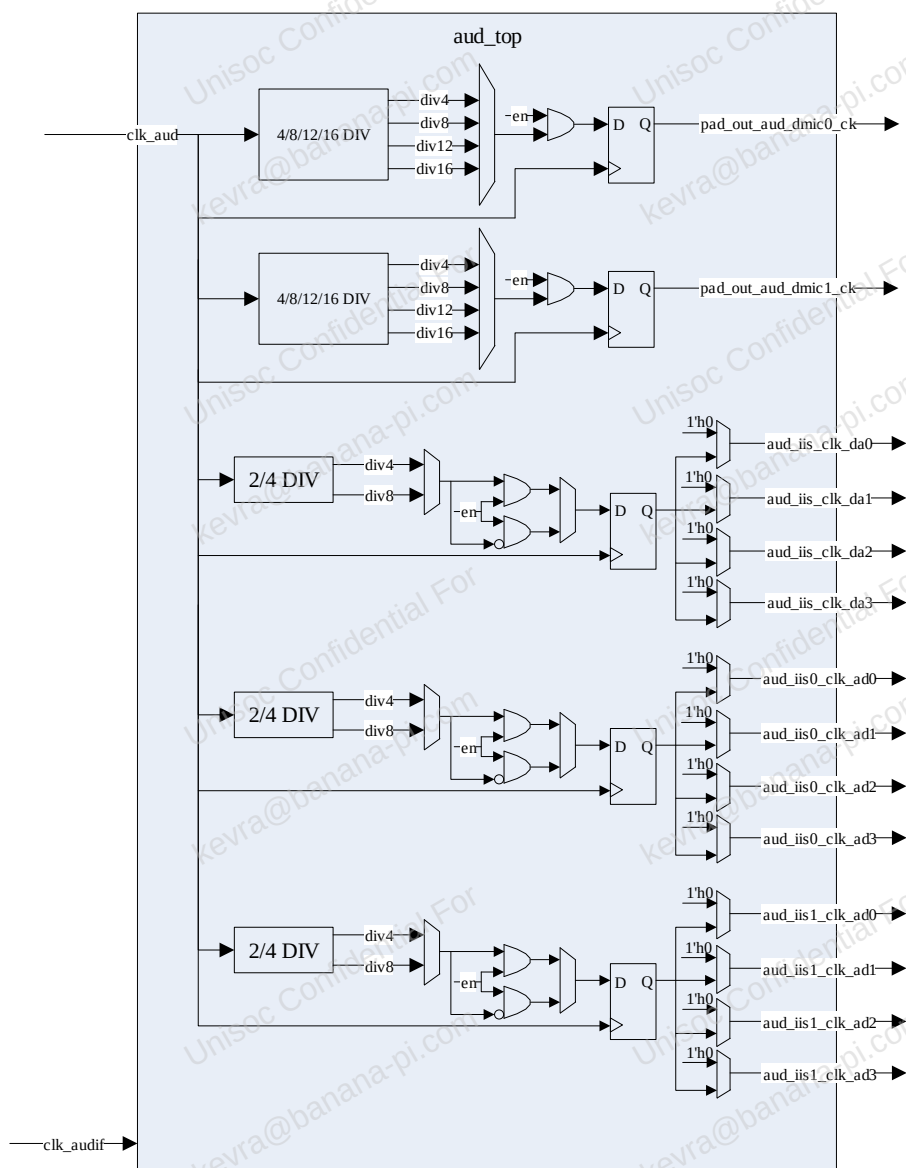
If the DNS function is enabled, the SDM input data will be selected from DNS output, otherwise from MUTE output data. PMIC analog PGA gain (ag_gain[3:0]) signals are sent to aud_ifd together with SDM output data. Another 1 bit PGA gain level signal(pga_lvl) is also output from DNS, which is used for simple DNS function designed in PMIC such as SC2730.

9.4.3.2 AUD clock structure

All AUD clock list in table below:

Clock	Direction	Source	Description
clk_aud	input	External SOC	26MHz, from external SOC
aud_iis_clk_da0	output	clk_aud	2/4 div of clk_aud
aud_iis_clk_da1	output	clk_aud	2/4 div of clk_aud
aud_iis_clk_da2	output	clk_aud	2/4 div of clk_aud
aud_iis_clk_da3	output	clk_aud	2/4 div of clk_aud
aud_iis0_clk_ad0	output	clk_aud	2/4 div of clk_aud
aud_iis0_clk_ad1	output	clk_aud	2/4 div of clk_aud
aud_iis0_clk_ad2	output	clk_aud	2/4 div of clk_aud
aud_iis0_clk_ad3	output	clk_aud	2/4 div of clk_aud
aud_iis1_clk_ad0	output	clk_aud	2/4 div of clk_aud
aud_iis1_clk_ad1	output	clk_aud	2/4 div of clk_aud
aud_iis1_clk_ad2	output	clk_aud	2/4 div of clk_aud
aud_iis1_clk_ad3	output	clk_aud	2/4 div of clk_aud
clk_audif	input	external SOC	26MHz/38.4MHz/51.2MHz, connect to PAD

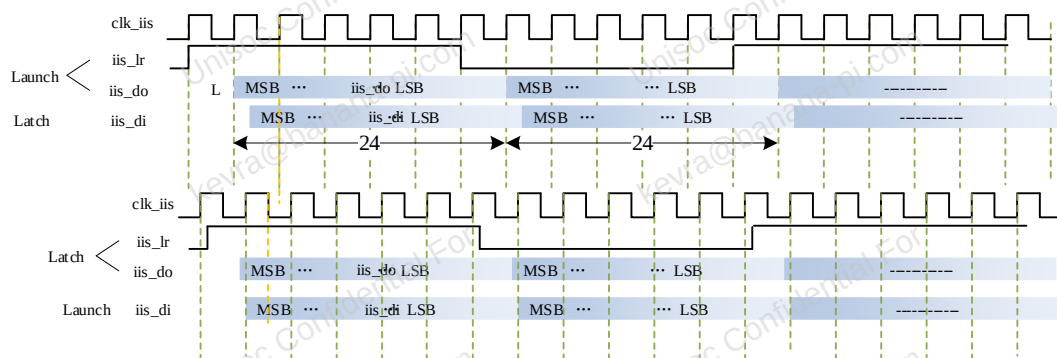
Figure 9-4 Audio clock structure



clk_aud is the main clock of AUD IP, the relationship with other clock shows in the figure above. All IIS clock are generated from clk_aud. clk_audif and clk_aud could set false path between each other.

The IIS in AUD designed as IIS master. The interface timing shows in figure below, iis_do and iis_lr launch at posedge of clk_iis in IIS master, and will be captured at negedge of clk_iis in IIS slave. iis_di is launch at posedge of clk_iis in IIS slave and capture at posedge clk_iis in AUD(IIS Master).

Figure 9-5 IIS interface timing



NOTE

launch in green line, capture in yellow line

9.4.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.4.5 Application Notes

DAC_MUTE User Guide

Software use 2 registers to control the mute functional behavior:

- **DAC_MUTE_EN**: dac mute function enable, when set 1, mute enable, when set 0, mute disable, this bit is controlled by software
- **DAC_MUTE_CTL**: software can use this bit to start mute, and stop mute when MP3 play, default is 0, when software set 1, a Gain will decrement from 4095 to 0, when Gain arrived to 0, MP3 will be muted to 0. When software set **DAC_MUTE_CTL** from 1 to 0, the Gain will increment from 0 to 4095.

Two timers are used to control the increment or decrement time step of the Gain, this two timer are controlled by **DAC_MUTE_EN**:

- **DAC_MUTE_DIV0_CTL**: first level timer threshold, software can configure these fields ,when timer counted to **DAC_MUTE_DIV0_CTL** value - 1, timer will restart to 0, the timer output valid will control the second level timer to start count. The first level timer count step is 26MHz cycle period.
- **DAC_MUTE_DIV1_CTL**: second level timer, software can configure these fields, when timer counted to **DAC_MUTE_DIV1_CTL** value -1, timer will restart to 0. This timer output valid will control the Gain increment or decrement.

9.5 VBC interface

9.5.1 Overview

The VBC module is designed as a digital interface, for software control and data handling with Audio Codec. Digital high-pass filters, digital gain, SRC, ALC,MBDRC, Notch Filter, Tone generator, 6 band EQ, 4 band EQ, side tone, Mixer, and loop path are also implemented in VBC for special applications.

9.5.2 Features

Clock

VBC main clock is 26MHz clock.

System bus

- One APB2 slave interface for DSP offload mp3 play, record. Voice call controlled by DSP, debug data information dump.
- One APB2 slave interface for AP audio play or record

Peripheral Interface

- Support 5 IIS slave interfaces
- Support 3 IIS master interface
- Support 1 TDM interface, can be configured to master or slave

Internal data path

- Support 4 dac path
- Support 8 adc path
- Support FM play and record path
- Support side tone path
- Support dac0 and dac1 loop to adc path
- Support 2 audio play and audio record path

Data FIFO for SW/DSP

- Support four 512×24 bits FIFO for dac path
- Support eight 512×24 bits FIFO for adc path
- Support two 256×24 bits FIFO for audio play path01
- Support two 128×24 bits FIFO for audio play path23
- Support two 512×24 bits FIFO which can send audio play data to DSP
- Support two 256×24 bits FIFO for 2 audio record path01
- Support two 128×24 bits FIFO for 2 audio record path23
- Support two 512×24 bits FIFO which send dsp record data to AP
- Programmable watermark for all FIFOs

General features for main function in data path

- DAC SRC is implemented for converting sample rate from 8/12/11.025/16/22.05/24/32/44.1 kHz to 48kHz
- ADC SRC is implemented for converting sample rate from 48kHz to 8/12/11.025/16/22.05/24/32/44.1 kHz
- Support Digital Gain control the volume of audio or voice
- Support Mixer that mixing left and right channel by different ways
- Support 6 band 3-order IIR filter can be used as High Pass Filter or EQ
- Support ALC to enhance the DAC path dynamic range
- Support cross talk for two DAC path

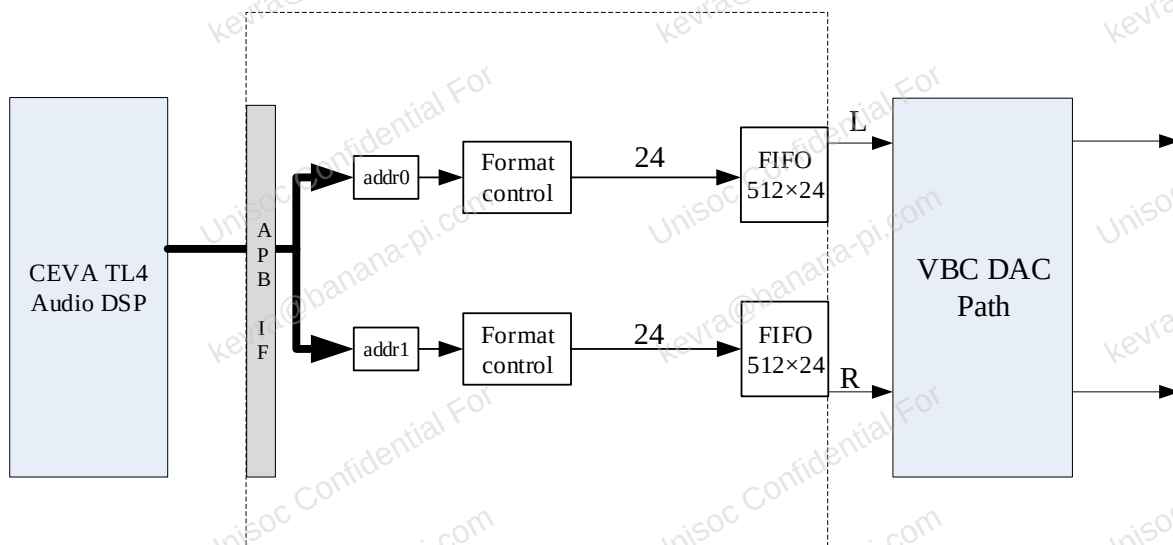
- Support 4 band EQ which coefficients can be configured dynamically.
- Support Multi-Band DRC to enhance dynamic range in high/middle/low band
- Support notch filter, vibrate/pilot tone generator, noise generator
- Support 16/24 bit data format for DAC path
- Support Limiter for DAC path
- Support Smooth Digital Gain control following ALC
- Support multiple channel Mixer for 2 DAC path with digital gain control for each channel
- Support digital gain with mute control for audio play and DSP play mixer

9.5.3 Function Description

9.5.3.1 DAC FIFO

- Audio processor can access this FIFO by APB bus
- APB bus is 32-bit width
- Two FIFO are implemented for each channel, FIFO depth is 512, data width is 24bits
- Support water mark control, FIFO full threshold and empty threshold can be programmed by software
- Support 24/16 bits' data format mode.
- Support stereo play, left and right channel have independent address for audio DSP access, after format control, left and right channel data will be filled into each FIFO

Figure 9-6 DAC FIFO



Data format control:

- 16 bit mode: APB wdata[31:0] only include 1 channel, low 16 bits is valid, for this format, data filled into FIFO should be expanded to 24 bits, {wdata[15:0], 8'h0}

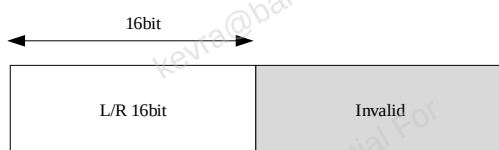
Figure 9-7 DAC FIFO Data format 1



32bit data in APB bus

- 16 bit mode: APB wdata[31:0] only include 1 channel, high 16 bits is valid, for this format, data filled into FIFO should be expanded to 24 bits, {wdata[31:16], 8'h0}

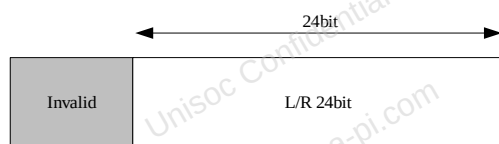
Figure 9-8 DAC FIFO Data format 2



32bit data in APB bus

- 24 bit mode: APB wdata[31:0] 1 channel, low 24 bit is valid, for this format, data filled into FIFO is wdata[23:0].

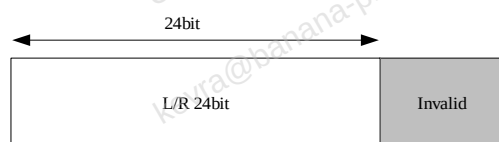
Figure 9-9 DAC FIFO Data format 3



32bit data in APB bus

- 24 bit mode: APB wdata[31:0] 1 channel, high 24 bit is valid, for this format, data filled into FIFO is left wdata[31:8].

Figure 9-10 DAC FIFO Data format 4



32bit data in APB bus

9.5.3.2 DAC SRC

The SRC in vbc dac data path is used to convert the sample rate from 8K/11.025K/12K/16K/22.05K/24K/32K/44.1K/48K to 48K.

- Support 8K/11.025K/12K/16K/22.05K/24K/32K/44.1K/48K input sample rate.
- Support change input sample rate when the module is idle.
- Support enable and disable SRC Limiter is 24 bit
- Support bypass interpolation filter
- Support 20-bit width symmetrical coefficient, not configurable
- Support two channel

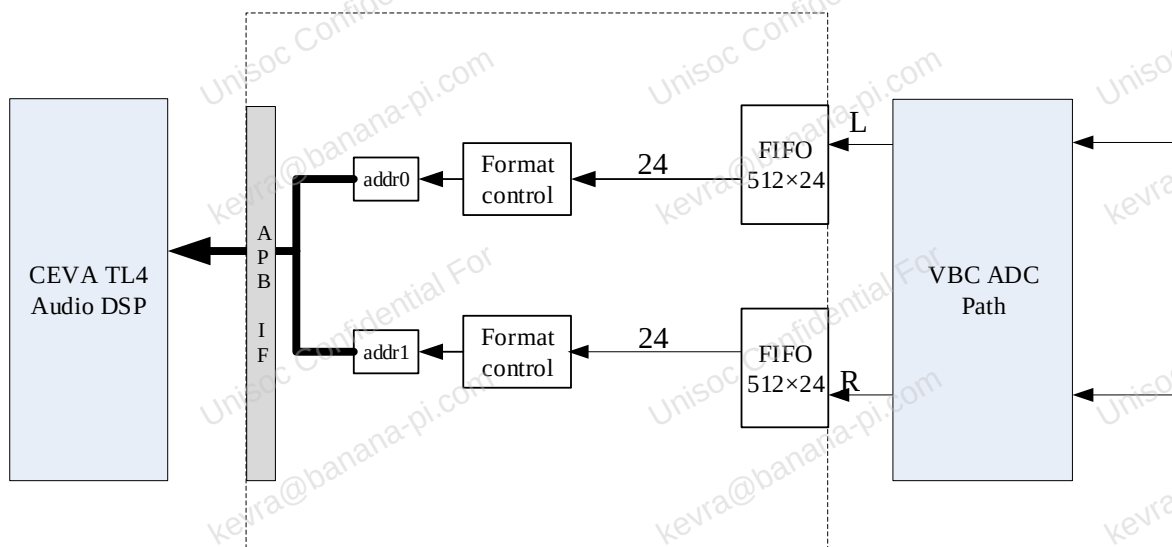
Table 9-5 DAC SRC mode

src_mode[3:0]	Sample rate
0	48kHz(bypass)
1	44.1kHz
2	32kHz
3	24kHz
4	22.05kHz
5	16kHz
6	12kHz
7	11.025kHz
8	9.6kHz(bypass)
9	8kHz

9.5.3.3 ADC FIFO

- Audio processor can access this FIFO by APB bus
- APB bus is 32-bit width
- Two FIFO are implemented for each channel,FIFO depth is 512, data width is 24bits
- Support water mark control, FIFO full threshold and empty threshold can be programmed by software
- Support 24/16 bits' data format mode

Figure 9-11 ADC FIFO



9.5.3.4 ADC SRC

The SRC in vbc dac data path is used to convert the sample rate from 48kHz to 8kHz/11.025kHz/12kHz/16kHz/22.05kHz/24kHz/32kHz/44.1kHz/48kHz.

- Support 48kHz input sample rate.
- Support change input sample rate when the module is idle.
- Support enable and disable SRC
- Limiter is 24 bit
- Support bypass interpolation filter
- Support 20-bit width symmetrical coefficient, not configurable

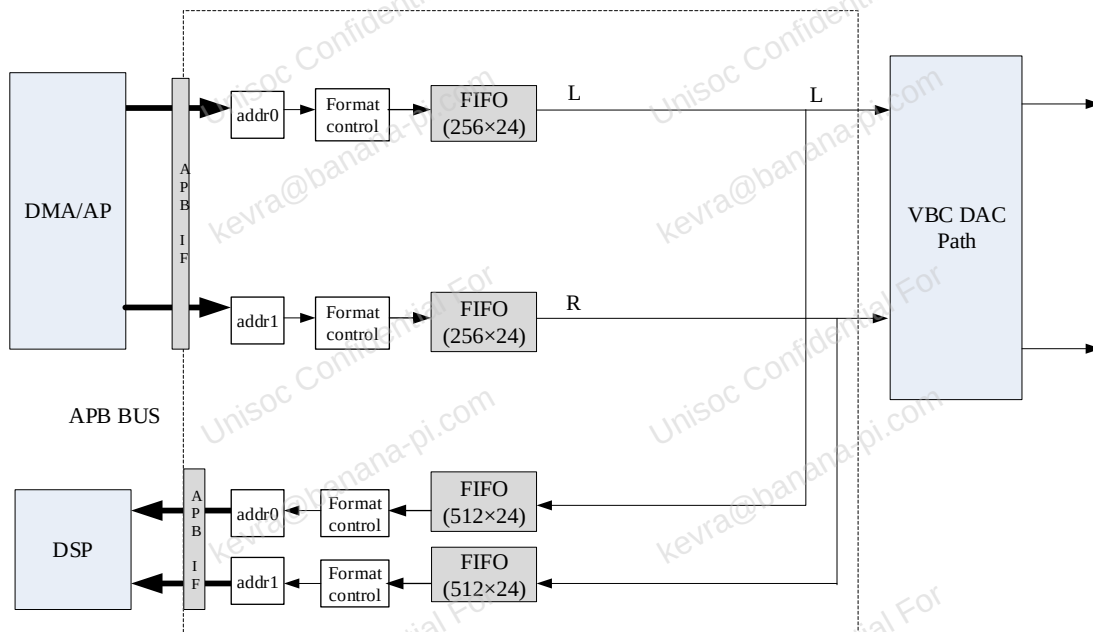
Table 9-6 ADC SRC mode

src_mode[3:0]	Sample rate
0	48kHz(bypass)
1	44.1kHz
2	32kHz
3	24kHz
4	22.05kHz
5	16kHz
6	12kHz
7	11.025kHz
8	9.6kHz(bypass)
9	8kHz

9.5.3.5 Audio Play PATH

- Support DMA transmission mechanism.
- Support independent DMA hardware request for left and right channel.
- Support independent FIFO for left and right channel. FIFO depth is 256, width is 24 bits
- Support two FIFO interface for CEVA TL4(Audio processor), the DSP can fetch data from these two FIFO. The FIFO depth is 512, width is 24 bits

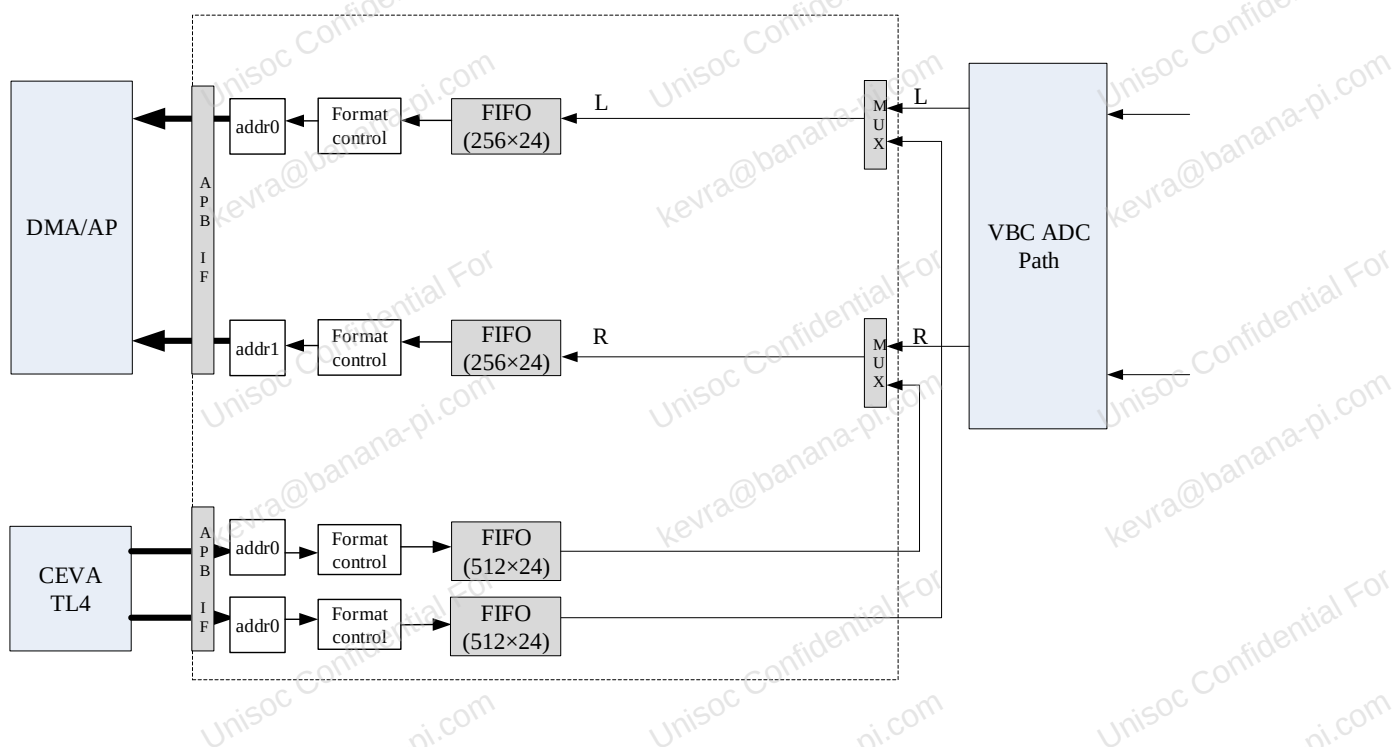
Figure 9-12 Audio Play FIFO



9.5.3.6 Audio Record PATH

- Support DMA transmission mechanism.
- Support independent DMA hardware request for left and right channel. Support independent FIFO for left and right channel. FIFO depth is 256, width is 24 bits
- Support FIFO interface for DSP(Audio processor), the DSP can fill data into FIFO, the FIFO depth is 512, width is 48 bits
- The format of audio record path data is the same audio play path data

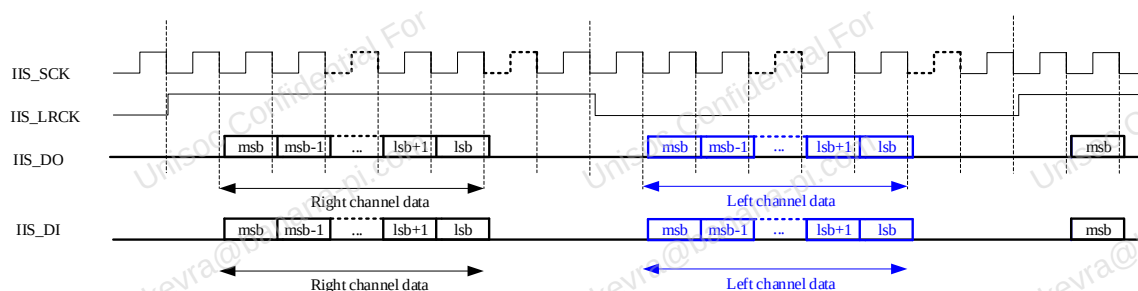
Figure 9-13 Audio Record FIFO



9.5.3.7 IIS Interface

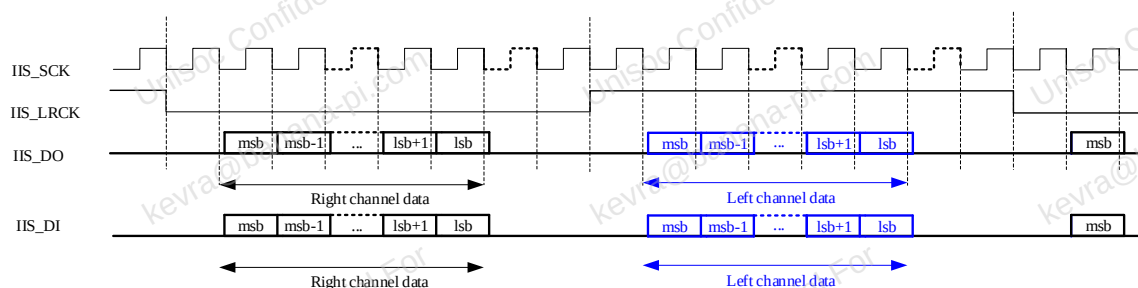
- Support master or slave mode
 - Support 24.576MHz IIS master work clock
 - Support IIS master sample rate: 48kHz/32kHz/24kHz/16kHz/8kHz
 - support 16/24-bit word length
 - support IIS msb-compatible format
 - support PCM short frame sync, two slot
1. IIS compatible mode, IIS LRCK initial low:

Figure 9-14 IIS timing 1



2. IIS compatible mode, IIS LRCK initial high:

Figure 9-15 IIS timing 2



3. PCM mode, short frame sync, 2 slot:

Figure 9-16 PCM timing

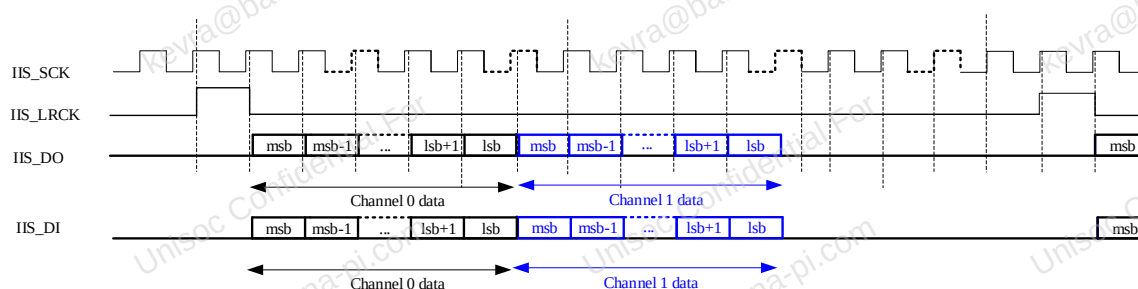


Table 9-7 IIS master mode configure

Sample rate	Bit rate	Channel Length	Divider
48kHz	3.072MHz	32	8
32kHz	2.048 MHz	32	12
24kHz	1.536 MHz	32	16
16kHz	1.024 MHz	32	24
8kHz	512kHz	32	48
48kHz	1.536MHz	16	16
32kHz	1.024MHz	16	24
24kHz	768 MHz	16	32
16kHz	512 MHz	16	48
8kHz	256kHz	16	96

9.5.3.8 TDM Interface

- Support master/slave mode
- Support 24.576MHz work clock

- Support 2, 4, 8 channel
- Support 48kHz/96kHz sample rate
- Support 16, 24 data width
- Support 16, 24, 32 slot width
- Support IIS compatible, LJ, RJ data format
- Support MSB, LSB mode
- Support tdm sync pulse width 50% duty cycle or one pulse
- Tdm receive data can be dispatched to DSP through 4 adc path, or one single FIFO.
- Support tdm work main clock with 24.576MHz when master mode.

Figure 9-17 TDM Connection

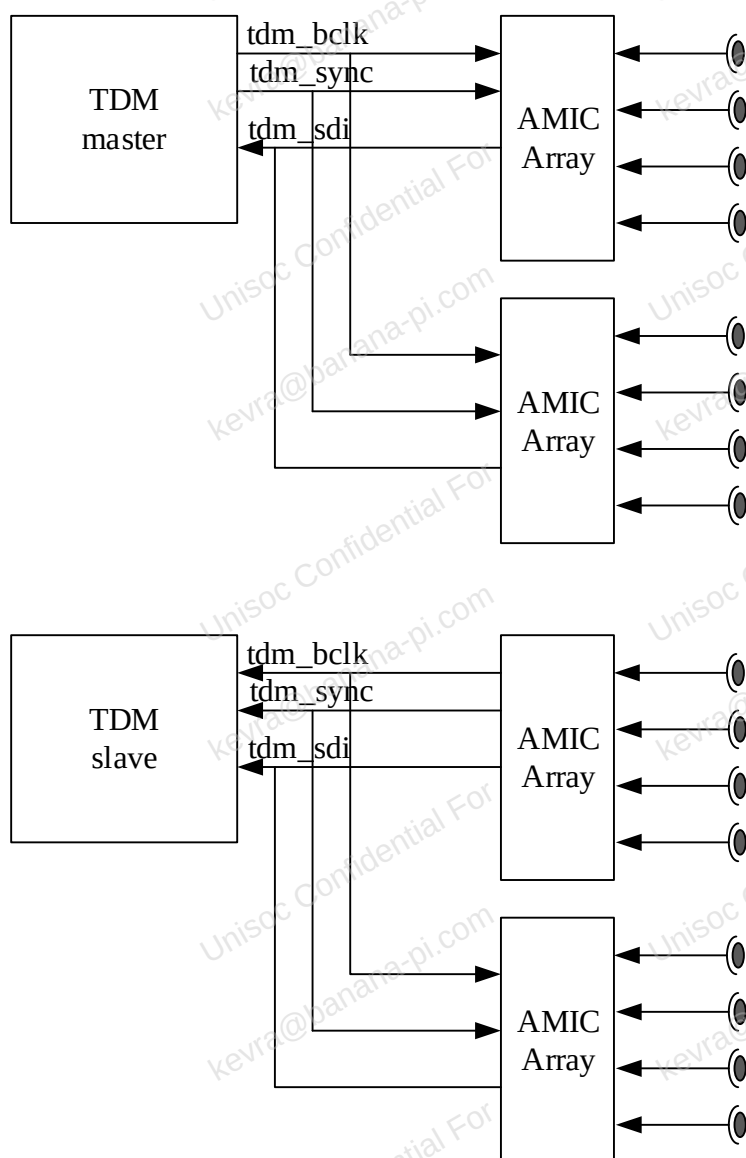
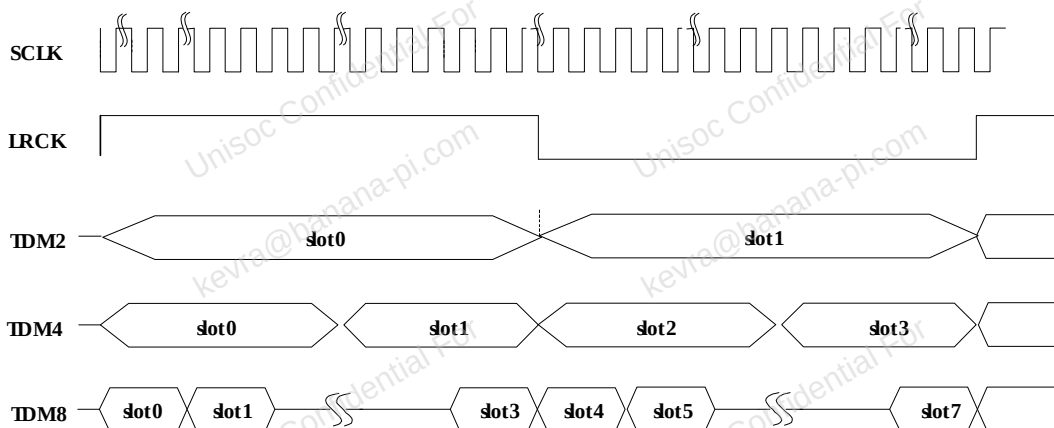


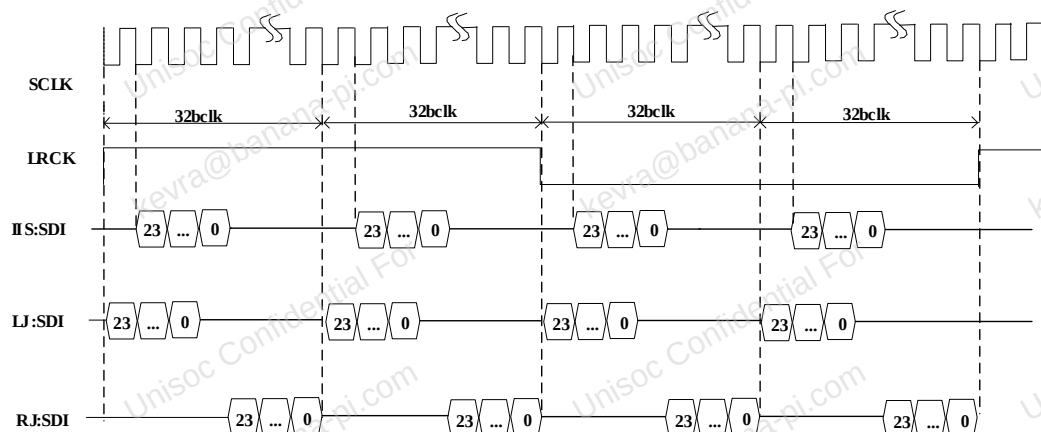
Table 9-8 TDM master mode configure

Channel	Sample rate	Bit rate	Channel Length	Divider
2	48kHz	1.536MHz	32	7
4	48kHz	3.072MHz	32	3
8	48kHz	6.144MHz	32	1
2	48kHz	3.072MHz	16	3
4	48kHz	6.144MHz	16	1
8	48kHz	12.288MHz	16	0
2	96kHz	3.072MHz	32	3
4	96kHz	6.144MHz	32	1
8	96kHz	12.288MHz	32	0
2	96kHz	6.144MHz	16	1
4	96kHz	12.288MHz	16	0
8	96kHz	24.576MHz	16	N/A

Figure 9-18 TDM 8/4/2 Channel timing


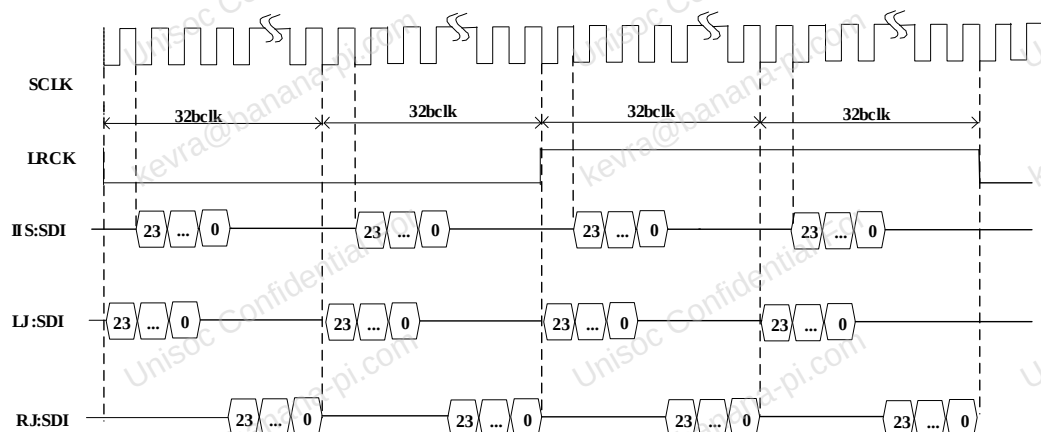
1. TDM4 timing 1(Master,Slave): data 24bit, slot 32bit, Compatible mode, LJ mode ,RJ mode

Figure 9-19 TDM 4 Channel timing 1



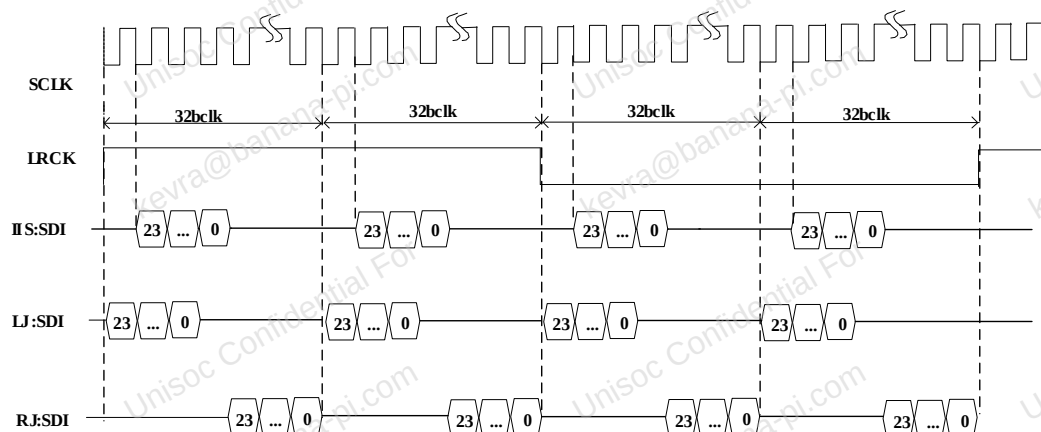
2. TDM4 timing 2(Master,Slave): data 24bit, slot 32bit, Compatible mode, LJ mode ,RJ mode

Figure 9-20 TDM 4 Channel timing 2



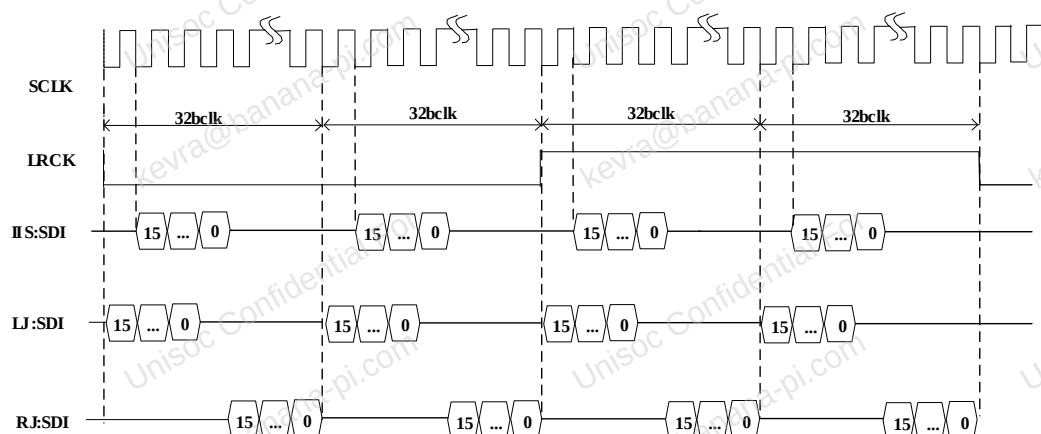
3. TDM4 timing 3(Master,Slave): data 16bit, slot 32bit, Compatible mode, LJ mode ,RJ mode

Figure 9-21 TDM 4 Channel timing 3



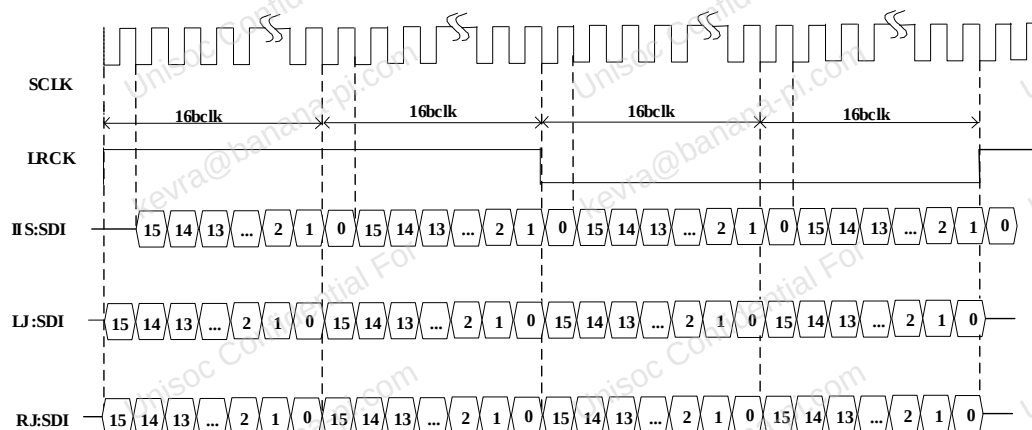
4. TDM4 timing 4(Master,Slave):data 16bit, slot 32bit, Compatible mode, LJ mode ,RJ mode

Figure 9-22 TDM 4 Channel timing 4



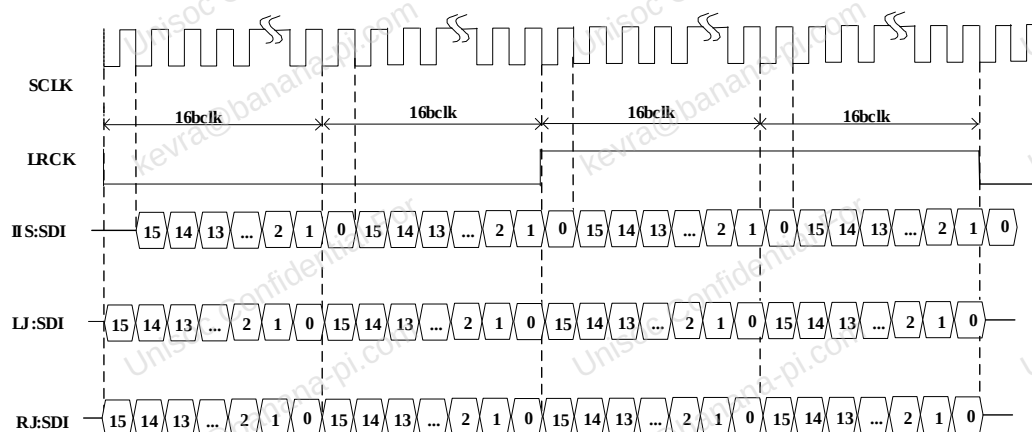
5. TDM4 timing 5(Master, Slave): data 16bit, slot 16bit, Compatible mode, LJ mode ,RJ mode

Figure 9-23 TDM 4 Channel timing 5



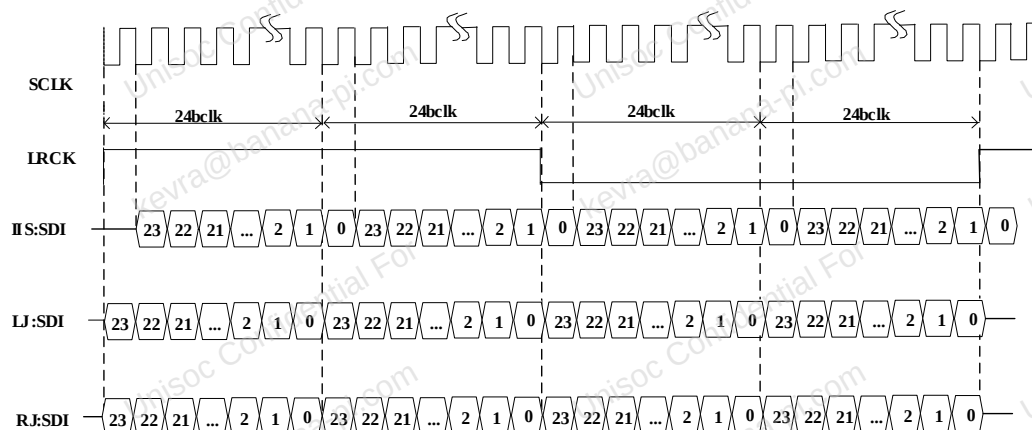
6. TDM4 timing 6(Master, Slave): data 16bit, slot 16bit, Compatible mode, LJ mode ,RJ mode

Figure 9-24 TDM 4 Channel timing 6



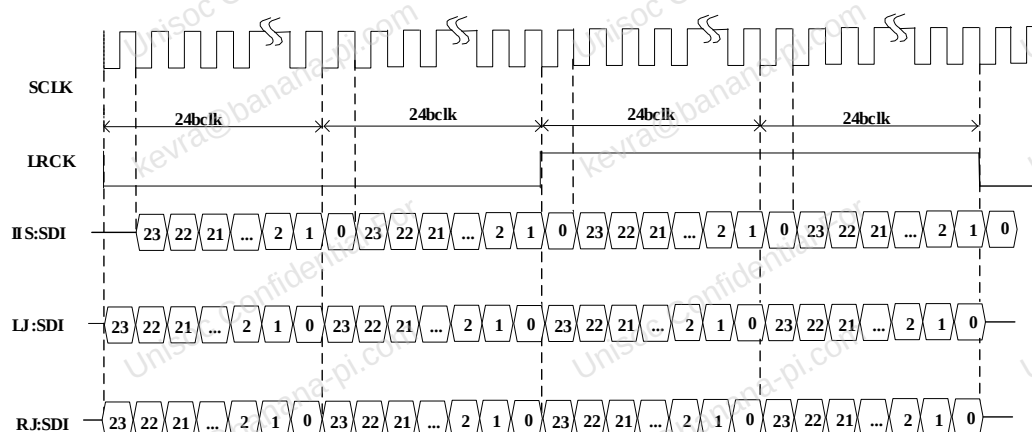
7. TDM4 timing 7(Slave only): data 24bit, slot 24bit, Compatible mode, LJ mode ,RJ mode

Figure 9-25 TDM 4 Channel timing 7



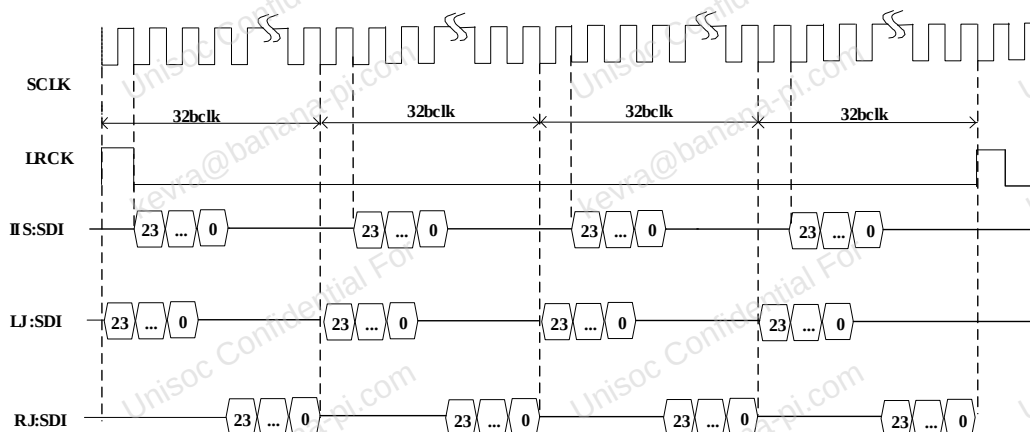
8. TDM4 timing 8(Slave only): data 24bit, slot 24bit, Compatible mode, LJ mode ,RJ mode

Figure 9-26 TDM 4 Channel timing 8



9. TDM4 timing 9(Master,Slave): pulse mode, data 24bit, slot 32bit, Compatible mode, LJ mode ,RJ mode

Figure 9-27 TDM 4 Channel timing 9



9.5.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.5.5 Application Notes

9.5.5.1 Announcements

When used VBC, there are some rules need to follow to avoid function error. Put it into two groups, a must group and a recommended group.

Must be followed:

- Before access ofld(AP) registers, firstly you must configure ofld_dp_en(0x0ED8: bit[7])
- Configure registers should not be changed (by software) when VBC is working, except special introduction.
- Before configure coefficient, related data path enable must be open firstly. For example, If you want to write coefficient into coefficient memory of DAC0 ALC, then dac0_dp_en(0x0ED8: bit[0]) should be configured firstly.
- When software-clear interrupt type is selected, the clear action must behind fill FIFO or read FIFO.
- Data input select registers of four ADC data path (0x0ECC: bit[11:9], bit[8:6], bit[5:3], bit[2:0]) must configure into different values.
- DAC data output select registers (0x0F0C: bit[3:2], bit[1:0]) must configure into different values.
- When Mp3 play from AP, SRC (sample rate convert) of DAC0 must be disabled.
- When adc3 data path is enabled, rf_fm_dp_en(0x0ED8:bit[7]) must be configured.
- When dac path is used (rf_dac1_bypass (0Xed8:bit[9]) is configured to 1), the signal rf_dac1_dp_en(0Xed8:bit[1]) must be configured to 1.
- Iis clk and lrck select signals(0Xec0, bit[0:13] bit[24:27]) must be configured before IIS clock enable signal (0Xf80 bit[2:7])

Recommended:

- Each software-oriented FIFO have a configurable water mark, between 1~511. But it is not recommended to set it too small or too large. Recommended value is 240.

- There are some modules of VBC which have one father enable and several children enable. It is recommended that children enable should be disabled when father enable is disabled.
- The SRC pause is not recommended to use
- Audio record FIFO depth is 256, and Audio record FIFO from DSP is 512, when Audio record data is from DSP, the DSP side FIFO almost empty threshold value shall less than 256, and data number that DSP put in shall not great than 256 in each interrupt.

9.5.5.2 Flow

9.5.5.2.1 Flow of Play

Play is a general concept, which not only refers to Mp3 play. Phone call, game voice, sounds of warning, they all need the Play flow.

Play flow can break into these steps:

1. Reset VBC: all clock domains needed. By this step, all registers and control logic are in the initial status.
2. Configure registers: after reset, you should configure the registers of VBC if you want it to work on. Registers configure mainly consist of open switches (data path enable and module enable), configure control registers, write coefficient into memory and so on. It also needs to follow a right step order to ensure the configuration is successful.
 - a Open dp_en: open data path enables, it controls the clock of sub-modules in the path. For play flow, mostly need to enable one or two or three of dac0, dac1, ofld data path enable.
 - b Coefficient: write coefficient in the coefficient memory, when the scenes include a module needs coefficient, then the step must be done here.
 - c General registers: configure the general registers except module enable, FIFO enable and interrupt enable.
 - d Module enable: enable the module used in the scenes.
 - e Clear: clear the modules used in the scenes. Some modules need to clear because it has a data memory which could not be initialised by reset; some modules which have internal FIFO may contains reserved data need to clear.
3. Pre-fill FIFO: fill data in the FIFO for play. The amount of data depends on the water mark configuration. It is recommended to fill the data just a little more than water-mark.
4. Enable interrupt/dma: open the interrupt enable or dma enable, depends on which type of data transfer is select.
5. Enable FIFO: open FIFO enable, then VBC will work on.
6. If codec IIS is master: Enable codec, initial audio codec, which contains the master of IIS enable.
7. If codec IIS is slave: initial Codec
8. If VBC IIS is master: enable VBC IIS master
9. Fill FIFO: if interrupt or DMA request is detected, fill data in the FIFO for play. The amount of data should not make the FIFO overflow, it means that the number you filled should be smaller than 512 minus the water-mark.

9.5.5.2.2 Flow of Record

Mostly the same with play.

1. Reset VBC: all clock domains needed. By this step, all registers and control logic are in the initial status.
2. Configure registers: after reset, you should configure the registers of VBC if you want it to work on. Registers configure mainly consist of open switches (data path enable and module enable), configure control

registers, write coefficient into memory and so on. It also needs to follow a right step order to ensure the configuration is successful.

- a Open dp_en: open data path enable, it controls the clock of sub-modules in the path. For record flow, it needs to enable adc data path enable.
 - b Coefficient: write coefficient in the coefficient memory, when the scenes include a module needs coefficient, then the step must be done here.
 - c General registers: configure the general registers except module enable, FIFO enable and interrupt enable.
 - d Module enable: enable the module used in the scenes.
 - e Clear: clear the modules used in the scenes. Some modules need to clear because it has a data memory which could not initial by reset; some modules which have internal FIFO may contain reserved data need to clear.
3. Enable interrupt/dma: open the interrupt enable or dma enable, depends on which data transfer type is select.
 4. Enable FIFO: open FIFO enable, let data write in.
 5. Enable codec: initialize audio codec, which contains IIS master.
 6. If VBC IIS master: enable IIS master
 7. If VBC tdm master: enable tdm master
 8. Read FIFO: if interrupt or DMA request is detected, read data out of the FIFO. The amount of data you read depends on the water-mark; it means that the number you read should not be larger than the water-mark.

9.5.5.2.3 Flow of FM Play

FM play flow can break into these steps:

1. Reset VBC: all clock domains needed. By this step, all registers and control logic are in the initial status.
2. Configure registers: after reset, you should configure the registers of VBC if you want it to work on. Registers configure mainly consist of open switches (data path enable and module enable), configure control registers, write coefficient into memory and so on. It also needs to follow a right step order to ensure the configuration is successful.
 - a Open dp_en: open data path enables, it controls the clock of sub-modules in the path. For FM play flow, mostly need to open one or two of dac0, dac1 data path enable, open FM data path enable.
 - b Coefficient: write coefficient in the coefficient memory, when the scenes include a module needs coefficient, then the step must be done here.
 - c General registers: configure the general registers except module enable, FIFO enable and interrupt enable.
 - d Module enable: enable the module used in the scenes.
 - e Clear: clear the modules used in the scenes. Some modules need to clear because it has a data memory which could not initial by reset; some modules which have internal FIFO may contain reserved data need to clear.
3. Pre-fill FIFO: fill some zero data (at least one) in the FIFO.
4. Enable FIFO: open DAC FIFO enable, then VBC will work on.
5. Enable codec: open audio codec, which contains the master of IIS.

9.5.5.3 Import module using specification

9.5.5.3.1 SRC

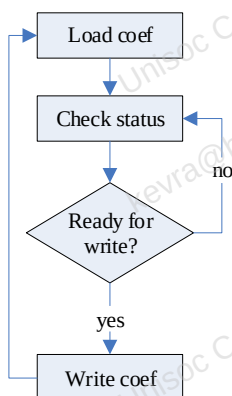
In VBC system, there are two types SRC(sample rate convert), one is ADC SRC which convert 48k sample rate to 8k/11.025k/12k/16k/22.05k/24k/32k/44.1k/48k, another is DAC SRC which convert

8k/11.025k/12k/16k/22.05k/24k/32k/44.1k/48k sample rate to 48k. Both of them have a data memory for data calculation during work time. To use SRC, you must initial the data memory with zero values and some control logic is already designed in SRC mode which could finish memory initial, but it needs a clear signal to active it. So between registers configuration, a soft-clear action should be done in the clear step.

A pause configure register is designed for software, but not recommend to use it because it will bring risk in.

9.5.5.3.2 Eq4

Eq4 is designed in DAC data path, which has four band, band0 and band2 support coefficient dynamic configuration. Dynamic configure means you can change the coefficient during Eq4 working time. When change the coefficient during working, the step should be followed show below:



- Load coef: there are two load types,
 - In the configure stage, after module enable step (the Play flow step 2-d), write register `rf_dac0_eq4_coef_updt` or `rf_dac1_eq4_coef_updt` to 1 to load coefficient;
 - During working time, after software write coefficient into the memory, write register `rf_dac0_eq4_coef_updt` or `rf_dac1_eq4_coef_updt` to 1 to load coefficient
- Check status: the status indicates that coefficient memory is available to write. Check the register (`dac0_eq4_coef_updt_sts` or `dac1_eq4_coef_updt_sts`) till gets a value 1, then write coefficient into the memory
- Write coef: write coefficient into the coefficient memory and write register `rf_dac0_eq4_coef_updt` or `rf_dac1_eq4_coef_updt` to 1 to load coefficient

There are some matters need to care:

- When don't use Eq4, keep the total enable (`rf_dac0_eq4_en` or `rf_dac1_eq4_en`) and all band enable (e.g. `rf_dac0_eq4_bd0_en`) disabled.
- Coefficient update should be as faster as you can, and if it take longer time than a period of 48k, then the whole DAC data path will be affect, several error data would be generated.

9.5.5.3.3 MBDRRC

When don't use MBDRRC, disable the father enable and all children enable.

9.5.5.3.4 IIS

IIS Slave timing configure

- IIS timing 1:
 - Iis rx*module registers setting:

```
#rf_vbc_iis*_lr_md = 0; // lrck default value 0, rising edge first
#rf_vbc_iis*_is_lsb = 0; //MSB mode, recommend use MSB mode.
#rf_vbc_iis*_pcm_md = 0; //not PCM mode, default IIS compatible mode
#rf_vbc_iis*_wd_sel = 0; //16 bit data Or #rf_vbc_iis*_wd_sel = 1; //24 bit data
```

- IIS tx* module registers setting:

```
#rf_dac*_iis_lr_md = 0;
#rf_dac*_iis_is_lsb = 0;
#rf_dac*_pcm_md = 0;
#rf_dac*_wd_sel = 0;
#rf_vbc_iis*_sck_inv = 0; // if data launch at falling edge of bit clock, set to 0, if data launch at rising edge of bit clock ,
set to 1;
```

- IIS timing 2:

- IIS rx*module registers setting:

```
#rf_vbc_iis*_lr_md = 1; // lrck default value 1, falling edge first
```

- IIS tx* module registers setting:

```
#rf_dac*_iis_lr_md = 1;
```

- PCM timing:

- IIS rx*module registers setting:

```
#rf_vbc_iis*_pcm_md = 1; // PCM mode
```

IIS Master timing configure

- IIS timing 1:

- IIS master module registers setting:

```
#rf_pcm_eb = 0;
#rf_iis_sck_rev = 0;
#rf_is_iisc_shrtf = 1; // IIS is compatible mode
#rf_iis_lrck_ph = 1; // lrck default 0, rising edge first
#rf_bpchn = 2'b01; // data is 16 bit
Or #rf_bpchn = 2'b11; // data is 32 bit, high 24 bit is valid
#rf_ext_inner_iis_mst_sel_* = 1; // lrck default 0, rising edge first
#rf_inside_iis_mst_en = 1; // IIS mst enable
```

- IIS rx*module registers setting:

```
#rf_vbc_iis*_lr_md = 1; // lrck default value 0, rising edge first
#rf_vbc_iis*_is_lsb = 0; //MSB mode, recommend use MSB mode.
#rf_vbc_iis*_pcm_md = 0; //not PCM mode, default IIS compatible mode
#rf_vbc_iis*_wd_sel = 0; //16 bit data
Or #rf_vbc_iis*_wd_sel = 1; //24 bit data
```

- IIS tx* module registers setting:

```
#rf_dac*_iis_lr_md = 0;
#rf_dac*_iis_is_lsb = 0;
#rf_dac*_pcm_md = 0;
#rf_dac*_wd_sel = 0;
```

```
#rf_vbc_iis*_sck_inv = 0; // if data launch at falling edge of bit clock, set to 0, if data launch at rising edge of bit clock ,
set to 1;
```

- IIS timing 2:

- IIS master module registers setting:

```
#rf_pcm_eb = 0;
#rf_iis_sck_rev = 0;
#rf_is_iisc_shrtf = 1; // IIS is compatible mode
#rf_iis_lrck_ph = 0; // lrck default 1, falling edge first
#rf_bpchn = 2'b10; // data is 16 bit
Or #rf_bpchn = 2'b11; // data is 32 bit, high 24 bit is valid
#rf_ext_inner_iis_mst_sel_* = 1; // lrck default 0, rising edge first
#rf_inside_iis_mst_en = 1; // IIS mst enable
```

- Iis rx*module registers setting:

```
#rf_vbc_iis*_lr_md = 1; // lrck default value 1, falling edge first
```

- Iis tx* module registers setting:

```
#rf_dac*_iis_lr_md = 1;
```

- PCM timing:

- IIS master module registers setting:

```
#rf_pcm_eb = 1;
#rf_iis_sck_rev = 1;
#rf_is_iisc_shrtf = 1; // IIS is compatible mode
#rf_iis_lrck_ph = 0; // lrck default 1, falling edge first
#rf_bpchn = 2'b10; // data is 16 bit
Or #rf_bpchn = 2'b11; // data is 32 bit, high 24 bit is valid
#rf_pcm_cyc = 2;
#rf_pcm_slot = 3'b001;
#rf_ext_inner_iis_mst_sel_* = 1; // lrck default 0, rising edge first
#rf_inside_iis_mst_en = 1; // IIS mst enable
```

- Iis rx*module registers setting:

```
#rf_vbc_iis*_pcm_md = 1; // PCM mode
```

TDM Slave timing configure

TDM slave module registers setting:

```
#rf_tdm_mst = 0; //tdm slave mode.
#rf_tdm_slot_num = 10; // 8 slot
#rf_tdm_slot_length = 0; // 32.
#rf_tdm_data_length = 0; // 24 bit data
#rf_tdm_iis_clkd = 0; // divider cnt
#rf_tdm_sample_rate = 0; // 48k
#rf_tdm_data_format = 0; // IIS compatible
#rf_tdm_lsb = 0; //MSB mode, recommend use MSB mode.
```

```
#rf_tdm_sck_inv = 0; // lrc change at positive edge of sck.
#rf_tdm_rx_edge = 1; // data receive at negative edge of sck.
#rf_tdm_en = 1; //24 bit data
```

TDM Master timing configure

TDM Master module registers setting:

```
#rf_tdm_mst = 1; //tdm master mode.
#rf_tdm_slot_num = 10; // 8 slot
#rf_tdm_slot_length = 0; // 32.
#rf_tdm_data_length = 0; // 24 bit data
#rf_tdm_iis_clkd = 0; // divider cnt
#rf_tdm_sample_rate = 0; // 48k
#rf_tdm_data_format = 0; // IIS compatible
#rf_tdm_lsb = 0; //MSB mode, recommend use MSB mode.
#rf_tdm_sck_inv = 0; // lrc change at positive edge of sck.
#rf_tdm_rx_edge = 1; // data receive at negative edge of sck.
#rf_tdm_en = 1; //24 bit data
```

IIS MUX/DE-MUX configure

Table 9-9 Adc paths IIS mux

path	Iis0	Iis1	Iis2	Iis3	miis
Adc0	Rf_iis0_sel=0 Rf_ext_inner_iis _mst_sel_0 =0	Rf_iis0_sel=1 Rf_ext_inner_iis _mst_sel_1 =0	Rf_iis0_sel=2 Rf_ext_inner_iis _mst_sel_2 =0	Rf_iis0_sel=3 Rf_ext_inner_iis _mst_sel_3 =0	Rf_iis0_sel=0 Rf_ext_inner_iis _mst_sel_0 =1 or Rf_iis0_sel=1 Rf_ext_inner_iis _mst_sel_1 =1 or Rf_iis0_sel=2 Rf_ext_inner_iis _mst_sel_2 =1 or Rf_iis0_sel=3 Rf_ext_inner_iis _mst_sel_3 =1

path	Iis0	Iis1	Iis2	Iis3	miis
Adc1	Rf_iis1_sel=0 Rf_ext_inner_iis _mst_sel_0 =0	Rf_iis1_sel=1 Rf_ext_inner_iis _mst_sel_1 =0	Rf_iis1_sel=2 Rf_ext_inner_iis _mst_sel_2 =0	Rf_iis1_sel=3 Rf_ext_inner_iis _mst_sel_3 =0	Rf_iis1_sel=0 Rf_ext_inner_iis _mst_sel_0 =1 or Rf_iis1_sel=1 Rf_ext_inner_iis _mst_sel_1 =1 or Rf_iis1_sel=2 Rf_ext_inner_iis _mst_sel_2 =1 or Rf_iis1_sel=3 Rf_ext_inner_iis _mst_sel_3 =1
Adc2	Rf_iis2_sel=0 Rf_ext_inner_iis _mst_sel_0 =0	Rf_iis2_sel=1 Rf_ext_inner_iis _mst_sel_1 =0	Rf_iis2_sel=2 Rf_ext_inner_iis _mst_sel_2 =0	Rf_iis2_sel=3 Rf_ext_inner_iis _mst_sel_3 =0	Rf_iis2_sel=0 Rf_ext_inner_iis _mst_sel_0 =1 or Rf_iis2_sel=1 Rf_ext_inner_iis _mst_sel_1 =1 or Rf_iis2_sel=2 Rf_ext_inner_iis _mst_sel_2 =1 or Rf_iis2_sel=3 Rf_ext_inner_iis _mst_sel_3 =1

path	Iis0	Iis1	Iis2	Iis3	miis
Adc3	Rf_iis3_sel=0 Rf_ext_inner_iis _mst_sel_0 =0	Rf_iis3_sel=1 Rf_ext_inner_iis _mst_sel_1 =0	Rf_iis3_sel=2 Rf_ext_inner_iis _mst_sel_2 =0	Rf_iis3_sel=3 Rf_ext_inner_iis _mst_sel_3 =0	Rf_iis3_sel=0 Rf_ext_inner_iis _mst_sel_0 =1 or Rf_iis3_sel=1 Rf_ext_inner_iis _mst_sel_1 =1 or Rf_iis3_sel=2 Rf_ext_inner_iis _mst_sel_2 =1 or Rf_iis3_sel=3 Rf_ext_inner_iis _mst_sel_3 =1

Table 9-10 Dac paths IIS clk and lrck select

Interface	Dac0	Dac1	Dac2
Iis0	Rf_dac0_path_sel=0 Rf_ext_inner_iis_mst_sel_0=0	Rf_dac1_path_sel=0 Rf_ext_inner_iis_mst_sel_0=0	Rf_dac2_path_sel=0 Rf_ext_inner_iis_mst_sel_0=0
Iis1	Rf_dac0_path_sel=1 Rf_ext_inner_iis_mst_sel_1=0	Rf_dac1_path_sel=1 Rf_ext_inner_iis_mst_sel_1=0	Rf_dac2_path_sel=1 Rf_ext_inner_iis_mst_sel_1=0
Iis2	Rf_dac0_path_sel=2 Rf_ext_inner_iis_mst_sel_2=0	Rf_dac1_path_sel=2 Rf_ext_inner_iis_mst_sel_2=0	Rf_dac2_path_sel=2 Rf_ext_inner_iis_mst_sel_2=0
Iis3	Rf_dac0_path_sel=3 Rf_ext_inner_iis_mst_sel_3=0	Rf_dac1_path_sel=3 Rf_ext_inner_iis_mst_sel_3=0	Rf_dac2_path_sel=3 Rf_ext_inner_iis_mst_sel_3=0
miis	Rf_dac0_path_sel=0 Rf_ext_inner_iis_mst_sel_0=1 Or Rf_dac0_path_sel=1 Rf_ext_inner_iis_mst_sel_1=1 Or Rf_dac0_path_sel=2 Rf_ext_inner_iis_mst_sel_2=1 Or Rf_dac0_path_sel=3 Rf_ext_inner_iis_mst_sel_3=1	Rf_dac1_path_sel=0 Rf_ext_inner_iis_mst_sel_0=1 Or Rf_dac1_path_sel=1 Rf_ext_inner_iis_mst_sel_1=1 Or Rf_dac1_path_sel=2 Rf_ext_inner_iis_mst_sel_2=1 Or Rf_dac1_path_sel=3 Rf_ext_inner_iis_mst_sel_3=1	Rf_dac2_path_sel=0 Rf_ext_inner_iis_mst_sel_0=1 Or Rf_dac2_path_sel=1 Rf_ext_inner_iis_mst_sel_1=1 Or Rf_dac2_path_sel=2 Rf_ext_inner_iis_mst_sel_2=1 Or Rf_dac2_path_sel=3 Rf_ext_inner_iis_mst_sel_3=1

Table 9-11 IIS do select

Interface	Dac0	Dac1	Dac2
Iis0_do	Rf_iis0_do_sel=0 Rf_ext_inner_iis_mst_sel_0=0	Rf_iis0_do_sel=1 Rf_ext_inner_iis_mst_sel_0=0	Rf_iis0_do_sel=2 Rf_ext_inner_iis_mst_sel_0=0
Iis1_do	Rf_iis1_do_sel=0 Rf_ext_inner_iis_mst_sel_0=0	Rf_iis1_do_sel=1 Rf_ext_inner_iis_mst_sel_0=0	Rf_iis1_do_sel=2 Rf_ext_inner_iis_mst_sel_0=0
Iis2_do	Rf_iis2_do_sel=0 Rf_ext_inner_iis_mst_sel_0=0	Rf_iis2_do_sel=1 Rf_ext_inner_iis_mst_sel_0=0	Rf_iis2_do_sel=2 Rf_ext_inner_iis_mst_sel_0=0
Iis3_do	Rf_iis3_do_sel=0 Rf_ext_inner_iis_mst_sel_0=0	Rf_iis3_do_sel=1 Rf_ext_inner_iis_mst_sel_0=0	Rf_iis3_do_sel=2 Rf_ext_inner_iis_mst_sel_0=0
Miis_do	Rf_iis0_do_sel=0 Rf_ext_inner_iis_mst_sel_0=1 Or Rf_iis1_do_sel=0 Rf_ext_inner_iis_mst_sel_1=1 or Rf_iis2_do_sel=0 Rf_ext_inner_iis_mst_sel_2=1 Or Rf_iis3_do_sel=0 Rf_ext_inner_iis_mst_sel_3=1	Rf_iis0_do_sel=1 Rf_ext_inner_iis_mst_sel_0=1 Or Rf_iis1_do_sel=1 Rf_ext_inner_iis_mst_sel_1=1 or Rf_iis2_do_sel=1 Rf_ext_inner_iis_mst_sel_2=1 Or Rf_iis3_do_sel=1 Rf_ext_inner_iis_mst_sel_3=1	Rf_iis0_do_sel=2 Rf_ext_inner_iis_mst_sel_0=1 Or Rf_iis1_do_sel=2 Rf_ext_inner_iis_mst_sel_1=1 or Rf_iis2_do_sel=2 Rf_ext_inner_iis_mst_sel_2=1 Or Rf_iis3_do_sel=2 Rf_ext_inner_iis_mst_sel_3=1

9.5.5.4 Scenario examples

9.5.5.4.1 Phone Call

Phone call is a play-flow scenario with side tone; one DAC data path and one ADC data path are used. In VBC, there are two stereo DAC paths and four stereo ADC paths for software, and which DAC path or ADC path you choose to use has no rules, depend on your needs. The example show below is a demo which uses DAC0 for play and ADC1 for side tone.

Follow the Play flow

1. Reset VBC.
2. Open DAC0 and ADC1 data path enable: 0x0ED8 bit[0]=1 and bit[3]=1.
3. Write coefficient into coefficient memory, offset address refer to *UIS7885 Audio System Control Registers Instructions*. If no module with coefficient used, ignore this step.
4. Select ADC1 data input IIS, choose IIS1, offset address 0x0EC0 bit[3:2]=1, offset address 0x0ECC bit[29]=0
5. Select ADC1 data input path, choose ADC1, offset address 0x0ECC bit[5:3]=1
6. Side tone select ADC1, offset address 0x0ECC bit[19:17]=3
7. Side tone mix DAC0 configure, offset address 0x0E70 bit[9:8]=2, bit[14:13]=2
8. Select DAC0 output IIS, choose IIS0, offset address 0x0EC0 bit[9:8]=0, offset address 0x0F0C bit[0]=0.

9. Configure IIS0 data width to 24bit, offset address 0x0EBC bit[26]=1
10. Open module enable of DAC0 and ADC1, Write offset address 0x0EDC, 0x0EE4 and 0x0EE8 relate bit to 1(depend on what module used).
11. Load Eq4 coefficient, offset address 0x0DF8 bit[0]=1(if you don't use Eq4, this step could ignore)
12. Clear all module you used, Write offset address 0x0DCC, 0xDD4 relate bit to 1(depend on what module used)
13. Pre-fill DAC0 FIFO, write left channel data into offset address 0x0000 and right channel data into offset address 0x0004, recommended amount of data are 240 sample points.
14. Open interrupt enable, offset address 0x0EA8 bit[1]=1, bit[3]=1
15. Open DAC0 FIFO enable, offset address 0x0F24 bit[0]=1, bit[1]=1
16. Open audio codec or extra codec.

9.5.5.4.2 Mp3 play

Mp3 play is a play-flow, when mp3 data comes from AP (ofld), only DAC0 data path could be used for play. Below is a simple demo for reference.

1. Reset VBC.
2. Open OFLD data path enable and DAC0 data path enable, offset address 0x0ED8 bit[0]=1, bit[6]=1
3. Write coefficient into coefficient memory, offset address refer to *UIS7885 Audio System Control Registers Instructions*. If no module with coefficient used, ignore this step.
4. Audio play data destination select, offset address 0x0F80 bit[0]=0
5. Disable DAC0 SRC, offset address 0x0EDC bit[0]=0
6. DAC0 play uses IIS1, offset address 0x0F0C bit[0]=0, offset address 0x0EC0 bit[9:8]=1, bit[11:10]=0
7. Configure IIS1 data width to 24bit, offset address 0x0EBC bit[27]=1
8. Open module enable of DAC0, Write offset address 0x0EDC and 0x0EE8 relate bit to 1(depend on what module used).
9. Load Eq4 coefficient, offset address 0x0DF8 bit[0]=1(if you don't use Eq4, this step could ignore)
10. Clear all module you used, Write offset address 0x0DCC relate bit to 1(depend on what module used)
11. Pre-fill DAC0 FIFO with some zero values (at least one zero), write zeros into offset address 0x0004 and 0x0000
12. Open SRC of OFLD is needed, offset address 0x0028 bit[3:0] for SRC mode, 0x002C bit[1:0] for SRC enable, write this two addresses from AP.
13. Pre-fill OFLD FIFO from AP, write Mp3 play data into offset address 0x0000 and 0x0004, recommended amount of data is 240 sample points.
14. Open audio play interrupt enable from AP, offset address 0x0044 bit[0]=1
15. Open audio play FIFO enable from AP, offset address 0x002C bit[9:8]=2'h3
16. Open DAC0 FIFO enable from DSP, offset address 0x0F24 bit[1:0]=2'h3
17. Open audio codec or extra codec.

9.5.5.4.3 FM play

Follow the flow of FM play, reference steps below for your FM application.

1. Reset VBC
2. Open FM data path enable and DAC1 data path enable, offset address 0x0ED8 bit[7]=1, bit[1]=1
3. Write coefficient into coefficient memory, offset address refer to *UIS7885 Audio System Control Registers Instructions*. If no module with coefficient used, ignore this step.

4. DAC1 play uses IIS1, offset address 0x0F0C bit[1]=0, offset address 0x0EC0 bit[9:8]=0 bit[11:10]=1
5. Configure IIS1 data width to 24bit, offset address 0x0EBC bit[27]=1
6. Open module enable of DAC1, write offset address 0x0EE0 and 0x0EE8 relate bit to 1(depend on what module used).
7. Load Eq4 coefficient, offset address 0x0DF8 bit[1]=1(if you don't use Eq4, this step could ignore)
8. Clear all module you used, Write offset address 0x0DD0 relate bit to 1(depend on what module used)
9. Pre-fill DAC1 FIFO with some zero values (at least one zero), write zeros into offset address 0x0008 and 0x000C
10. FM mix DAC1 configure, offset address 0x0E64 bit[9:8]=1, bit[16:15]=1
11. If FM SRC is used, be careful of FM SRC out ready select. If no FM data is record to DSP or AP, offset address 0x0ECC bit[2:0] != 3, bit[5:3] != 3, bit[8:6] != 3, bit[11:9] != 3.
12. Configure to use interface IIS3, offset address 0x0EC0 bit[7:6]=3
13. Configure IIS data width to 24 bit, offset address 0x0EBC bit[15]=1
14. Open DAC1 FIFO enable, offset address 0x0F24 bit[3:2]=2'h3
15. Open audio codec or extra codec.
16. Open extra FM

9.5.5.4.4 Audio record to DSP

Follow the flow of Record, reference steps below for your record application

1. Reset VBC
2. Open ADC2 data path enable, offset address 0x0ED8 bit[4]=1
3. Write coefficient into coefficient memory, offset address refer to *UIS7885 Audio System Control Registers Instructions*. If no module with coefficient used, ignore this step.
4. Audio record uses IIS2, offset address 0x0EC0 bit[5:4]=2
5. Configure IIS data width to 24 bit, offset address 0x0EBC bit[14]
6. Configure ADC2 data input select, offset address 0x0ECC bit[8:6]=2
7. Open module enable of ADC2, write offset address 0x0EE4 relate bit to 1(depend on what module used).
8. Clear all module you used, write offset address 0x0DD4 relate bit to 1(depend on what module used)
9. Enable ADC2 interrupt, offset address 0x0EA8 bit[5]=1
10. Enable ADC2 FIFO, offset address 0x0F24 bit[9:8]=2'h3
11. Open audio codec or extra codec

9.5.5.4.5 Audio record to AP

Compared with the record flow to DSP, record to AP need more configurations on AP registers. A simple demo shows below.

1. Reset VBC
2. Open ADC0 data path enable and OFLD data path enable, offset address 0x0ED8 bit[2]=1, bit[6]=1
3. Write coefficient into coefficient memory, offset address refer to *UIS7885 Audio System Control Registers Instructions*. If no module with coefficient used, ignore this step.
4. Audio record uses IIS1, offset address 0x0EC0 bit[1:0]=1
5. Configure IIS data width to 24bit, offset address 0x0EBC bit[12]=1
6. Configure ADC0 data input select, offset address 0x0ECC bit[2:0]=0
7. Configure ADC data select registers, offset address 0x0ECC bit[15:14]=2'h0

8. Ensure audio record DSP FIFO enable is disabled, offset address 0x0F24 bit[15:14]=0
9. Open module enable of ADC0, write offset address 0x0EE4 relate bit to 1(depend on what module used).
10. Clear all module you used, write offset address 0x0DD4 relate bit to 1(depend on what module used)
11. Enable audio record SRC if needed, offset address 0x002C bit[3:2]=2'h3, SRC mode offset address 0x0028 bit[7:4]
12. Enable audio record AP interrupt from AP, offset address 0x0044 bit[1]=1.
13. Open audio codec or extra codec

9.5.5.4.6 Voice mix

There are two mix points of DAC0 and DAC1, one is after module SMTH_DG, another is in module MIXDG. First mix point only has digital gain for local data, do not have digital gain for extra data; second mix point has digital gain of the both.

- 0x0DE4 bit[15:0] is digital gain of DAC0 left channel data
- 0x0DE4 bit[31:16] is digital gain of DAC0 right channel data
- 0x0F78 bit[15:0] is digital gain of extra DAC1 left channel data
- 0x0F78 bit[31:16] is digital gain of extra DAC1 right channel data
- 0x0DE8 bit[15:0] is digital gain of DAC1 left channel data
- 0x0DE8 bit[31:16] is digital gain of DAC1 right channel data
- 0x0F7C bit[15:0] is digital gain of extra DAC0 left channel data
- 0x0F7C bit[31:16] is digital gain of extra DAC0 right channel data

9.5.5.5 Coefficient memory mapping

9.5.5.5.1 DAC0_EQ6_ALC_COEF memory mapping

Offset address	Name	Description	Width
0x0050	DAC0_HPCOEF0	High pass coef0	24
0x0054	DAC0_HPCOEF1	High pass coef1	24
0x0058	DAC0_HPCOEF2	High pass coef2	24
0x005C	DAC0_HPCOEF3	High pass coef3	24
0x0060	DAC0_HPCOEF4	High pass coef4	24
0x0064	DAC0_HPCOEF5	High pass coef5	24
0x0068	DAC0_HPCOEF6	High pass coef6	24
0x006C	DAC0_HPCOEF7	High pass coef7	24
0x0070	DAC0_HPCOEF8	High pass coef8	24
0x0074	DAC0_HPCOEF9	High pass coef9	24
0x0078	DAC0_HPCOEF10	High pass coef10	24
0x007C	DAC0_HPCOEF11	High pass coef11	24
0x0080	DAC0_HPCOEF12	High pass coef12	24

Offset address	Name	Description	Width
0x0084	DAC0_HPCOEF13	High pass coef13	24
0x0088	DAC0_HPCOEF14	High pass coef14	24
0x008C	DAC0_HPCOEF15	High pass coef15	24
0x0090	DAC0_HPCOEF16	High pass coef16	24
0x0094	DAC0_HPCOEF17	High pass coef17	24
0x0098	DAC0_HPCOEF18	High pass coef18	24
0x009C	DAC0_HPCOEF19	High pass coef19	24
0x00A0	DAC0_HPCOEF20	High pass coef20	24
0x00A4	DAC0_HPCOEF21	High pass coef21	24
0x00A8	DAC0_HPCOEF22	High pass coef22	24
0x00AC	DAC0_HPCOEF23	High pass coef23	24
0x00B0	DAC0_HPCOEF24	High pass coef24	24
0x00B4	DAC0_HPCOEF25	High pass coef25	24
0x00B8	DAC0_HPCOEF26	High pass coef26	24
0x00BC	DAC0_HPCOEF27	High pass coef27	24
0x00C0	DAC0_HPCOEF28	High pass coef28	24
0x00C4	DAC0_HPCOEF29	High pass coef29	24
0x00C8	DAC0_HPCOEF30	High pass coef30	24
0x00CC	DAC0_HPCOEF31	High pass coef31	24
0x00D0	DAC0_HPCOEF32	High pass coef32	24
0x00D4	DAC0_HPCOEF33	High pass coef33	24
0x00D8	DAC0_HPCOEF34	High pass coef34	24
0x00DC	DAC0_HPCOEF35	High pass coef35	24
0x00E0	DAC0_HPCOEF36	High pass coef36	24
0x00E4	DAC0_HPCOEF37	High pass coef37	24
0x00E8	DAC0_HPCOEF38	High pass coef38	24
0x00EC	DAC0_HPCOEF39	High pass coef39	24
0x00F0	DAC0_HPCOEF40	High pass coef40	24
0x00F4	DAC0_HPCOEF41	High pass coef41	24
0x00F8	DAC0_HPCOEF42	High pass coef42	24
0x00FC	DAC0_ALC_RISE	Alc rise coef	16

Offset address	Name	Description	Width
0x0100	DAC0_ALC_FALL	Alc fall coef	16
0x0104	DAC0_ALC_LMT	Alc limit coef	16
0x0108	DAC0_ALC_THD	Alc thd coef	16
0x010C	DAC0_ALC_RATIO	Alc ratio coef	16
0x0110	DAC0_ALC_CG_VAR	Alc cg variate coef	16
0x0114	DAC0_ALC_ATK_RATE	Alc atk rate coef	16
0x0118	DAC0_ALC_ATK_RATE_EX	Alc atk rate external coef	16
0x011C	DAC0_ALC_HOLD	Alc hold coef	16
0x0120	DAC0_ALC_RLS_RATE	Alc RLS rate coef	16
0x0124	DAC0_ALC_RLS_RATE_EX	Alc RLS rate external coef	16
0x0128	DAC0_ALC_EXTEND_LMT	Alc extend LMT coef	16
0x012C	DAC0_ALC_EXTEND_THD	Alc extend THD coef	16
0x0130	DAC0_ALC_EXTEND_RATIO	Alc extend ratio coef	16
0x0134	DAC0_ALC_EXTEND_HOLD	Alc extend hold coef	16
0x0138	DAC0_ALC_EXTEND_LEVEL_L	Reserved	16
0x013C	DAC0_ALC_EXTEND_LEVEL_H	Reserved	16

9.5.5.5.2 DAC0_EQ4_COEF memory mapping

Offset address	Name	Description	Width
0x0140	DAC0_EQ4_COEF0	Anti_Resonant_gain	24
0x0144	DAC0_EQ4_COEF1	Anti_Resonant_b(1)	24
0x0148	DAC0_EQ4_COEF2	Anti_Resonant_b(2)	24
0x014C	DAC0_EQ4_COEF3	Anti_Resonant_b(3)	24
0x0150	DAC0_EQ4_COEF4	Anti_Resonant_a(2)	24
0x0154	DAC0_EQ4_COEF5	Anti_Resonant_a(3)	24
0x0158	DAC0_EQ4_COEF6	LCF_gain	24
0x015C	DAC0_EQ4_COEF7	LCF_b (1)	24
0x0160	DAC0_EQ4_COEF8	LCF_b (2)	24
0x0164	DAC0_EQ4_COEF9	LCF_b (3)	24
0x0168	DAC0_EQ4_COEF10	LCF_b (2)	24
0x016C	DAC0_EQ4_COEF11	LCF_b (3)	24

Offset address	Name	Description	Width
0x0170	DAC0_EQ4_COEF12	BWE_gain	24
0x0174	DAC0_EQ4_COEF13	BWE_b (1)	24
0x0178	DAC0_EQ4_COEF14	BWE_b (2)	24
0x017C	DAC0_EQ4_COEF15	BWE_b (3)	24
0x0180	DAC0_EQ4_COEF16	BWE_b (2)	24
0x0184	DAC0_EQ4_COEF17	BWE_b (3)	24
0x0188	DAC0_EQ4_COEF18	HS_gain	24
0x018C	DAC0_EQ4_COEF19	HS_b (1)	24
0x0190	DAC0_EQ4_COEF20	HS_b (2)	24
0x0194	DAC0_EQ4_COEF21	HS_b (3)	24
0x0198	DAC0_EQ4_COEF22	HS_b (2)	24
0x019C	DAC0_EQ4_COEF23	HS_b (3)	24

9.5.5.5.3 DAC0_MBDRC_BLEQ_DRC_COEFmemory mapping

Offset address	Name	Description	Width
0x01A0	DAC0_MBDRC_BL_EQ1_COEF0	Left channel BL_HPFS1_gain	24
0x01A4	DAC0_MBDRC_BL_EQ1_COEF1	Left channel BL_HPFS1_b(1)	24
0x01A8	DAC0_MBDRC_BL_EQ1_COEF2	Left channel BL_HPFS1_b(2)	24
0x01AC	DAC0_MBDRC_BL_EQ1_COEF3	Left channel BL_HPFS1_b(3)	24
0x01B0	DAC0_MBDRC_BL_EQ1_COEF4	Left channel BL_HPFS1_b (2)	24
0x01B4	DAC0_MBDRC_BL_EQ1_COEF5	Left channel BL_HPFS1_b (3)	24
0x01B8	DAC0_MBDRC_BL_EQ1_COEF6	Right channel BL_HPFS1_gain	24
0x01BC	DAC0_MBDRC_BL_EQ1_COEF7	Right channel BL_HPFS1_b(1)	24
0x01C0	DAC0_MBDRC_BL_EQ1_COEF8	Right channel BL_HPFS1_b(2)	24
0x01C4	DAC0_MBDRC_BL_EQ1_COEF9	Right channel BL_HPFS1_b(3)	24
0x01C8	DAC0_MBDRC_BL_EQ1_COEF10	Right channel BL_HPFS1_b (2)	24
0x01CC	DAC0_MBDRC_BL_EQ1_COEF11	Right channel BL_HPFS1_b (3)	24
0x01D0	DAC0_BL_DRC_RISE	BL DRC rise coef	16
0x01D4	DAC0_BL_DRC_FALL	BL DRC fall coef	16
0x01D8	DAC0_BL_DRC_LMT	BL DRC limit coef	16

Offset address	Name	Description	Width
0x01DC	DAC0_BL_DRC_THD	BL DRC THD coef	16
0x01E0	DAC0_BL_DRC_RATIO	BL DRC ratio coef	16
0x01E4	DAC0_BL_DRC_CG_VAR	BL DRC CG variate coef	16
0x01E8	DAC0_BL_DRC_ATK_RATE	BL DRC ATK rate coef	16
0x01EC	DAC0_BL_DRC_ATK_RATE_EX	BL DRC ATK rate external coef	16
0x01F0	DAC0_BL_DRC_HOLD	BL DRC hold coef	16
0x01F4	DAC0_BL_DRC_RLS_RATE	BL DRC RLS rate coef	16
0x01F8	DAC0_BL_DRC_RLS_RATE_EX	BL DRC RLS rate external coef	16
0x01FC	DAC0_BL_DRC_EXTEND_LMT	BL DRC extend LMT coef	16
0x0200	DAC0_BL_DRC_EXTEND_THD	BL DRC extend THD coef	16
0x0204	DAC0_BL_DRC_EXTEND_RATIO	BL DRC extend ratio coef	16
0x0208	DAC0_BL_DRC_EXTEND_HOLD	BL DRC extend hold coef	16
0x020C	DAC0_BL_DRC_EXTEND_LEVEL_L	BL DRC extend level coef of left channel	16
0x0210	DAC0_BL_DRC_EXTEND_LEVEL_H	BL DRC extend level coef of right channel	16

9.5.5.5.4 DAC0_MBDRC_CROSS_EQ_COEF memory mapping

Offset address	Name	Description	Width
0x0214	DAC0_MBDRC_HIGH_EQ2_COEF0	Left channel HIGH_HPFS2_gain	24
0x0218	DAC0_MBDRC_HIGH_EQ2_COEF1	Left channel HIGH_HPFS2_b(1)	24
0x021C	DAC0_MBDRC_HIGH_EQ2_COEF2	Left channel HIGH_HPFS2_b(2)	24
0x0220	DAC0_MBDRC_HIGH_EQ2_COEF3	Left channel HIGH_HPFS2_b(3)	24
0x0224	DAC0_MBDRC_HIGH_EQ2_COEF4	Left channel HIGH_HPFS2_b (2)	24
0x0228	DAC0_MBDRC_HIGH_EQ2_COEF5	Left channel HIGH_HPFS2_b (3)	24
0x022C	DAC0_MBDRC_HIGH_EQ2_COEF6	Right channel HIGH_HPFS2_gain	24
0x0230	DAC0_MBDRC_HIGH_EQ2_COEF7	Right channel HIGH_HPFS2_b(1)	24
0x0234	DAC0_MBDRC_HIGH_EQ2_COEF8	Right channel HIGH_HPFS2_b(2)	24
0x0238	DAC0_MBDRC_HIGH_EQ2_COEF9	Right channel HIGH_HPFS2_b(3)	24
0x023C	DAC0_MBDRC_HIGH_EQ2_COEF10	Right channel HIGH_HPFS2_b (2)	24
0x0240	DAC0_MBDRC_HIGH_EQ2_COEF11	Right channel HIGH_HPFS2_b (3)	24

Offset address	Name	Description	Width
0x0244	DAC0_MBDRC_HIGH_EQ2_COEF12	Left channel HIGH_HPF_s3_gain	24
0x0248	DAC0_MBDRC_HIGH_EQ2_COEF13	Left channel HIGH_HPF_s3_b(1)	24
0x024C	DAC0_MBDRC_HIGH_EQ2_COEF14	Left channel HIGH_HPF_s3_b(2)	24
0x0250	DAC0_MBDRC_HIGH_EQ2_COEF15	Left channel HIGH_HPF_s3_b(3)	24
0x0254	DAC0_MBDRC_HIGH_EQ2_COEF16	Left channel HIGH_HPF_s3_b (2)	24
0x0258	DAC0_MBDRC_HIGH_EQ2_COEF17	Left channel HIGH_HPF_s3_b (3)	24
0x025C	DAC0_MBDRC_HIGH_EQ2_COEF18	Right channel HIGH_HPF_s3_gain	24
0x0260	DAC0_MBDRC_HIGH_EQ2_COEF19	Right channel HIGH_HPF_s3_b(1)	24
0x0264	DAC0_MBDRC_HIGH_EQ2_COEF20	Right channel HIGH_HPF_s3_b(2)	24
0x0268	DAC0_MBDRC_HIGH_EQ2_COEF21	Right channel HIGH_HPF_s3_b(3)	24
0x027C	DAC0_MBDRC_HIGH_EQ2_COEF22	Right channel HIGH_HPF_s3_b (2)	24
0x0270	DAC0_MBDRC_HIGH_EQ2_COEF23	Right channel HIGH_HPF_s3_b (3)	24
0x0274	DAC0_MBDRC_MID_EQ4_COEF0	Left channel MID_HPF_s4_gain	24
0x0278	DAC0_MBDRC_MID_EQ4_COEF1	Left channel MID_HPF_s4_b(1)	24
0x027C	DAC0_MBDRC_MID_EQ4_COEF2	Left channel MID_HPF_s4_b(2)	24
0x0280	DAC0_MBDRC_MID_EQ4_COEF3	Left channel MID_HPF_s4_b(3)	24
0x0284	DAC0_MBDRC_MID_EQ4_COEF4	Left channel MID_HPF_s4_b (2)	24
0x0288	DAC0_MBDRC_MID_EQ4_COEF5	Left channel MID_HPF_s4_b (3)	24
0x028C	DAC0_MBDRC_MID_EQ4_COEF6	Right channel MID_HPF_s4_gain	24
0x0290	DAC0_MBDRC_MID_EQ4_COEF7	Right channel MID_HPF_s4_b(1)	24
0x0294	DAC0_MBDRC_MID_EQ4_COEF8	Right channel MID_HPF_s4_b(2)	24
0x0298	DAC0_MBDRC_MID_EQ4_COEF9	Right channel MID_HPF_s4_b(3)	24
0x029C	DAC0_MBDRC_MID_EQ4_COEF10	Right channel MID_HPF_s4_b (2)	24
0x02A0	DAC0_MBDRC_MID_EQ4_COEF11	Right channel MID_HPF_s4_b (3)	24
0x02A4	DAC0_MBDRC_MID_EQ4_COEF12	Left channel MID_HPF_s5_gain	24
0x02A8	DAC0_MBDRC_MID_EQ4_COEF13	Left channel MID_HPF_s5_b(1)	24
0x02AC	DAC0_MBDRC_MID_EQ4_COEF14	Left channel MID_HPF_s5_b(2)	24
0x02B0	DAC0_MBDRC_MID_EQ4_COEF15	Left channel MID_HPF_s5_b(3)	24
0x02B4	DAC0_MBDRC_MID_EQ4_COEF16	Left channel MID_HPF_s5_b (2)	24
0x02B8	DAC0_MBDRC_MID_EQ4_COEF17	Left channel MID_HPF_s5_b (3)	24
0x02BC	DAC0_MBDRC_MID_EQ4_COEF18	Right channel MID_HPF_s5_gain	24

Offset address	Name	Description	Width
0x02C0	DAC0_MBDRC_MID_EQ4_COEF19	Right channel MID_HPF_s5_b(1)	24
0x02C4	DAC0_MBDRC_MID_EQ4_COEF20	Right channel MID_HPF_s5_b(2)	24
0x02C8	DAC0_MBDRC_MID_EQ4_COEF21	Right channel MID_HPF_s5_b(3)	24
0x02CC	DAC0_MBDRC_MID_EQ4_COEF22	Right channel MID_HPF_s5_b (2)	24
0x02D0	DAC0_MBDRC_MID_EQ4_COEF23	Right channel MID_HPF_s5_b (3)	24
0x02D4	DAC0_MBDRC_MID_EQ4_COEF24	Left channel MID_HPF_s6_gain	24
0x02D8	DAC0_MBDRC_MID_EQ4_COEF25	Left channel MID_HPF_s6_b(1)	24
0x02DC	DAC0_MBDRC_MID_EQ4_COEF26	Left channel MID_HPF_s6_b(2)	24
0x02E0	DAC0_MBDRC_MID_EQ4_COEF27	Left channel MID_HPF_s6_b(3)	24
0x02E4	DAC0_MBDRC_MID_EQ4_COEF28	Left channel MID_HPF_s6_b (2)	24
0x02E8	DAC0_MBDRC_MID_EQ4_COEF29	Left channel MID_HPF_s6_b (3)	24
0x02EC	DAC0_MBDRC_MID_EQ4_COEF30	Right channel MID_HPF_s6_gain	24
0x02F0	DAC0_MBDRC_MID_EQ4_COEF31	Right channel MID_HPF_s6_b(1)	24
0x02F4	DAC0_MBDRC_MID_EQ4_COEF32	Right channel MID_HPF_s6_b(2)	24
0x02F8	DAC0_MBDRC_MID_EQ4_COEF33	Right channel MID_HPF_s6_b(3)	24
0x02FC	DAC0_MBDRC_MID_EQ4_COEF34	Right channel MID_HPF_s6_b (2)	24
0x0300	DAC0_MBDRC_MID_EQ4_COEF35	Right channel MID_HPF_s6_b (3)	24
0x0304	DAC0_MBDRC_MID_EQ4_COEF36	Left channel MID_HPF_s7_gain	24
0x0308	DAC0_MBDRC_MID_EQ4_COEF37	Left channel MID_HPF_s7_b(1)	24
0x030C	DAC0_MBDRC_MID_EQ4_COEF38	Left channel MID_HPF_s7_b(2)	24
0x0310	DAC0_MBDRC_MID_EQ4_COEF39	Left channel MID_HPF_s7_b(3)	24
0x0314	DAC0_MBDRC_MID_EQ4_COEF40	Left channel MID_HPF_s7_b (2)	24
0x0318	DAC0_MBDRC_MID_EQ4_COEF41	Left channel MID_HPF_s7_b (3)	24
0x031C	DAC0_MBDRC_MID_EQ4_COEF42	Right channel MID_HPF_s7_gain	24
0x0320	DAC0_MBDRC_MID_EQ4_COEF43	Right channel MID_HPF_s7_b(1)	24
0x0324	DAC0_MBDRC_MID_EQ4_COEF44	Right channel MID_HPF_s7_b(2)	24
0x0328	DAC0_MBDRC_MID_EQ4_COEF45	Right channel MID_HPF_s7_b(3)	24
0x032C	DAC0_MBDRC_MID_EQ4_COEF46	Right channel MID_HPF_s7_b (2)	24
0x0330	DAC0_MBDRC_MID_EQ4_COEF47	Right channel MID_HPF_s7_b (3)	24
0x0334	DAC0_MBDRC_LOW_EQ2_0_COEF0	Left channel HIGH_HPF_s8_gain	24
0x0338	DAC0_MBDRC_LOW_EQ2_0_COEF1	Left channel HIGH_HPF_s8_b(1)	24

Offset address	Name	Description	Width
0x033C	DAC0_MBDRC_LOW_EQ2_0_COEF2	Left channel HIGH_HPFS8_b(2)	24
0x0340	DAC0_MBDRC_LOW_EQ2_0_COEF3	Left channel HIGH_HPFS8_b(3)	24
0x0344	DAC0_MBDRC_LOW_EQ2_0_COEF4	Left channel HIGH_HPFS8_b (2)	24
0x0348	DAC0_MBDRC_LOW_EQ2_0_COEF5	Left channel HIGH_HPFS8_b (3)	24
0x034C	DAC0_MBDRC_LOW_EQ2_0_COEF6	Right channel HIGH_HPFS8_gain	24
0x0350	DAC0_MBDRC_LOW_EQ2_0_COEF7	Right channel HIGH_HPFS8_b(1)	24
0x0354	DAC0_MBDRC_LOW_EQ2_0_COEF8	Right channel HIGH_HPFS8_b(2)	24
0x0358	DAC0_MBDRC_LOW_EQ2_0_COEF9	Right channel HIGH_HPFS8_b(3)	24
0x035C	DAC0_MBDRC_LOW_EQ2_0_COEF10	Right channel HIGH_HPFS8_b (2)	24
0x0360	DAC0_MBDRC_LOW_EQ2_0_COEF11	Right channel HIGH_HPFS8_b (3)	24
0x0364	DAC0_MBDRC_LOW_EQ2_0_COEF12	Left channel HIGH_HPFS9_gain	24
0x0368	DAC0_MBDRC_LOW_EQ2_0_COEF13	Left channel HIGH_HPFS9_b(1)	24
0x036C	DAC0_MBDRC_LOW_EQ2_0_COEF14	Left channel HIGH_HPFS9_b(2)	24
0x0370	DAC0_MBDRC_LOW_EQ2_0_COEF15	Left channel HIGH_HPFS9_b(3)	24
0x0374	DAC0_MBDRC_LOW_EQ2_0_COEF16	Left channel HIGH_HPFS9_b (2)	24
0x0378	DAC0_MBDRC_LOW_EQ2_0_COEF17	Left channel HIGH_HPFS9_b (3)	24
0x037C	DAC0_MBDRC_HIGH_EQ2_COEF18	Right channel HIGH_HPFS9_gain	24
0x0380	DAC0_MBDRC_LOW_EQ2_0_COEF19	Right channel HIGH_HPFS9_b(1)	24
0x0384	DAC0_MBDRC_LOW_EQ2_0_COEF20	Right channel HIGH_HPFS9_b(2)	24
0x0388	DAC0_MBDRC_LOW_EQ2_0_COEF21	Right channel HIGH_HPFS9_b(3)	24
0x038C	DAC0_MBDRC_LOW_EQ2_0_COEF22	Right channel HIGH_HPFS9_b (2)	24
0x0390	DAC0_MBDRC_LOW_EQ2_0_COEF23	Right channel HIGH_HPFS9_b (3)	24

9.5.5.5.5 DAC0_MBDRC_CROS_DRC_COEF memory mapping

Offset address	Name	Description	Width
0x0394	DAC0_HIGH_DRC_RISE	High DRC rise coef	16
0x0398	DAC0_HIGH_DRC_FALL	High DRC fall coef	16
0x039C	DAC0_HIGH_DRC_LMT	High DRC LMT coef	16
0x03A0	DAC0_HIGH_DRC_THD	High DRC THD coef	16
0x03A4	DAC0_HIGH_DRC_RATIO	High DRC ratio coef	16

Offset address	Name	Description	Width
0x03A8	DAC0_HIGH_DRC_CG_VAR	High DRC CG variate coef	16
0x03AC	DAC0_HIGH_DRC_ATK_RATE	High DRC atk rate coef	16
0x03B0	DAC0_HIGH_DRC_ATK_RATE_EX	High DRC atk rate external coef	16
0x03B4	DAC0_HIGH_DRC_HOLD	High DRC hold coef	16
0x03B8	DAC0_HIGH_DRC_RLS_RATE	High DRC RLS rate coef	16
0x03BC	DAC0_HIGH_DRC_RLS_RATE_EX	High DRC RLS rate external coef	16
0x03C0	DAC0_HIGH_DRC_EXTEND_LMT	High DRC extend LMT coef	16
0x03C4	DAC0_HIGH_DRC_EXTEND_THD	High DRC extend THD coef	16
0x03C8	DAC0_HIGH_DRC_EXTEND_RATIO	High DRC extend ratio coef	16
0x03CC	DAC0_HIGH_DRC_EXTEND_HOLD	High DRC extend hold coef	16
0x03D0	DAC0_HIGH_DRC_EXTEND_LEVEL_L	High DRC extend level coef of left channel	16
0x03D4	DAC0_HIGH_DRC_EXTEND_LEVEL_H	High DRC extend level coef of right channel	16
0x03D8	DAC0_MID_DRC_RISE	MID DRC rise coef	16
0x03DC	DAC0_MID_DRC_FALL	MID DRC fall coef	16
0x03E0	DAC0_MID_DRC_LMT	MID DRC LMT coef	16
0x03E4	DAC0_MID_DRC_THD	MID DRC THD coef	16
0x03E8	DAC0_MID_DRC_RATIO	MID DRC ratio coef	16
0x03EC	DAC0_MID_DRC_CG_VAR	MID DRC cg variate coef	16
0x03F0	DAC0_MID_DRC_ATK_RATE	MID DRC ATK rate coef	16
0x03F4	DAC0_MID_DRC_ATK_RATE_EX	MID DRC ATK rate external coef	16
0x03F8	DAC0_MID_DRC_HOLD	MID DRC hold coef	16
0x03FC	DAC0_MID_DRC_RLS_RATE	MID DRC RLS rate coef	16
0x0400	DAC0_MID_DRC_RLS_RATE_EX	MID DRC RLS rate external coef	16
0x0404	DAC0_MID_DRC_EXTEND_LMT	MID DRC extend LMT coef	16
0x0408	DAC0_MID_DRC_EXTEND_THD	MID DRC extend THD coef	16
0x040C	DAC0_MID_DRC_EXTEND_RATIO	MID DRC extend ratio coef	16
0x0410	DAC0_MID_DRC_EXTEND_HOLD	MID DRC extend hold coef	16
0x0414	DAC0_MID_DRC_EXTEND_LEVEL_L	MID DRC extend level coef of left channel	16
0x0418	DAC0_MID_DRC_EXTEND_LEVEL_H	MID DRC extend level of right channel	16

Offset address	Name	Description	Width
0x041C	DAC0_LOW_DRC_RISE	LOW DRC rise coef	16
0x0420	DAC0_LOW_DRC_FALL	LOW DRC fall coef	16
0x0424	DAC0_LOW_DRC_LMT	LOW DRC LMT coef	16
0x0428	DAC0_LOW_DRC_THD	LOW DRC THD coef	16
0x042C	DAC0_LOW_DRC_RATIO	LOW DRC ratio coef	16
0x0430	DAC0_LOW_DRC_CG_VAR	LOW DRC cg variate coef	16
0x0434	DAC0_LOW_DRC_ATK_RATE	LOW DRC ATK rate coef	16
0x0438	DAC0_LOW_DRC_ATK_RATE_EX	LOW DRC ATK rate external coef	16
0x043C	DAC0_LOW_DRC_HOLD	LOW DRC hold coef	16
0x0440	DAC0_LOW_DRC_RLS_RATE	LOW DRC RLS rate coef	16
0x0444	DAC0_LOW_DRC_RLS_RATE_EX	LOW DRC RLS rate external coef	16
0x0448	DAC0_LOW_DRC_EXTEND_LMT	LOW DRC extend LMT coef	16
0x044C	DAC0_LOW_DRC_EXTEND_THD	LOW DRC extend THD coef	16
0x0450	DAC0_LOW_DRC_EXTEND_RATIO	LOW DRC extend ratio coef	16
0x0454	DAC0_LOW_DRC_EXTEND_HOLD	LOW DRC extend hold coef	16
0x0458	DAC0_LOW_DRC_EXTEND_LEVEL_L	LOW DRC extend level coef of left channel	16
0x045C	DAC0_LOW_DRC_EXTEND_LEVEL_H	LOW DRC extend level coef of right channel	16
0x0460	DAC0_POST_DRC_RISE	POST DRC rise coef	16
0x0464	DAC0_POST_DRC_FALL	POST DRC fall coef	16
0x0468	DAC0_POST_DRC_LMT	POST DRC LMT coef	16
0x046C	DAC0_POST_DRC_THD	POST DRC THD coef	16
0x0470	DAC0_POST_DRC_RATIO	POST DRC ratio coef	16
0x0474	DAC0_POST_DRC_CG_VAR	POST DRC CG variate coef	16
0x0478	DAC0_POST_DRC_ATK_RATE	POST DRC ATK rate coef	16
0x047C	DAC0_POST_DRC_ATK_RATE_EX	POST DRC ATK rate external coef	16
0x0480	DAC0_POST_DRC_HOLD	POST DRC hold coef	16
0x0484	DAC0_POST_DRC_RLS_RATE	POST DRC RLS rate coef	16
0x0488	DAC0_POST_DRC_RLS_RATE_EX	POST DRC RLS rate external coef	16
0x048C	DAC0_POST_DRC_EXTEND_LMT	POST DRC extend LMT coef	16

Offset address	Name	Description	Width
0x0490	DAC0_POST_DRC_EXTEND_THD	POST DRC extend coef	16
0x0494	DAC0_POST_DRC_EXTEND_RATIO	POST DRC extend ratio coef	16
0x0498	DAC0_POST_DRC_EXTEND_HOLD	POST DRC extend hold coef	16
0x049C	DAC0_POST_DRC_EXTEND_LEVEL_L	POST DRC extend level coef of left channel	16
0x04A0	DAC0_POST_DRC_EXTEND_LEVEL_H	POST DRC extend level coef of right channel	16

9.5.5.5.6 DAC0_NCH_COEF memory mapping

Offset address	Name	Description	Width
0x04A4	DAC0_MBDRC_LOW_EQ2_1_COEF0	Left channel LOW_HPFS10_gain	24
0x04A8	DAC0_MBDRC_LOW_EQ2_1_COEF1	Left channel LOW_HPFS10_b(1)	24
0x04AC	DAC0_MBDRC_LOW_EQ2_1_COEF2	Left channel LOW_HPFS10_b(2)	24
0x04B0	DAC0_MBDRC_LOW_EQ2_1_COEF3	Left channel LOW_HPFS10_b(3)	24
0x04B4	DAC0_MBDRC_LOW_EQ2_1_COEF4	Left channel LOW_HPFS10_b (2)	24
0x04B8	DAC0_MBDRC_LOW_EQ2_1_COEF5	Left channel LOW_HPFS10_b (3)	24
0x04BC	DAC0_MBDRC_LOW_EQ2_1_COEF6	Right channel LOW_HPFS10_gain	24
0x04C0	DAC0_MBDRC_LOW_EQ2_1_COEF7	Right channel LOW_HPFS10_b(1)	24
0x04C4	DAC0_MBDRC_LOW_EQ2_1_COEF8	Right channel LOW_HPFS10_b(2)	24
0x04C8	DAC0_MBDRC_LOW_EQ2_1_COEF9	Right channel LOW_HPFS10_b(3)	24
0x04CC	DAC0_MBDRC_LOW_EQ2_1_COEF10	Right channel LOW_HPFS10_b (2)	24
0x04D0	DAC0_MBDRC_LOW_EQ2_1_COEF11	Right channel LOW_HPFS10_b (3)	24
0x04D4	DAC0_MBDRC_LOW_EQ2_1_COEF12	Left channel LOW_HPFS11_gain	24
0x04D8	DAC0_MBDRC_LOW_EQ2_1_COEF13	Left channel LOW_HPFS11_b(1)	24
0x04DC	DAC0_MBDRC_LOW_EQ2_1_COEF14	Left channel LOW_HPFS11_b(2)	24
0x04E0	DAC0_MBDRC_LOW_EQ2_1_COEF15	Left channel LOW_HPFS11_b(3)	24
0x04E4	DAC0_MBDRC_LOW_EQ2_1_COEF16	Left channel LOW_HPFS11_b (2)	24
0x04E8	DAC0_MBDRC_LOW_EQ2_1_COEF17	Left channel LOW_HPFS11_b (3)	24
0x04EC	DAC0_MBDRC_LOW_EQ2_1_COEF18	Right channel LOW_HPFS11_gain	24
0x04F0	DAC0_MBDRC_LOW_EQ2_1_COEF19	Right channel LOW_HPFS11_b(1)	24
0x04F4	DAC0_MBDRC_LOW_EQ2_1_COEF20	Right channel LOW_HPFS11_b(2)	24

Offset address	Name	Description	Width
0x04F8	DAC0_MBDRC_LOW_EQ2_1_COEF21	Right channel LOW_HPF_s11_b(3)	24
0x04FC	DAC0_MBDRC_LOW_EQ2_1_COEF22	Right channel LOW_HPF_s11_b (2)	24
0x0500	DAC0_MBDRC_LOW_EQ2_1_COEF23	Right channel LOW_HPF_s11_b (3)	24
0x0504	DAC0_NCH_COEF0	Left channel Elliptic4_HPF_s1_gain_L	24
0x0508	DAC0_NCH_COEF1	Left channel Elliptic4_HPF_s1_b_L(1)	24
0x050C	DAC0_NCH_COEF2	Left channel Elliptic4_HPF_s1_b_L (2)	24
0x0510	DAC0_NCH_COEF3	Left channel Elliptic4_HPF_s1_b_L (3)	24
0x0514	DAC0_NCH_COEF4	Left channel Elliptic4_HPF_s1_b_L (2)	24
0x0518	DAC0_NCH_COEF5	Left channel Elliptic4_HPF_s1_b_L (3)	24
0x051C	DAC0_NCH_COEF6	Right channel Elliptic4_HPF_s1_gain_R	24
0x0520	DAC0_NCH_COEF7	Right channel Elliptic4_HPF_s1_b_R(1)	24
0x0524	DAC0_NCH_COEF8	Right channel Elliptic4_HPF_s1_b_R (2)	24
0x0528	DAC0_NCH_COEF9	Right channel Elliptic4_HPF_s1_b_R (3)	24
0x052C	DAC0_NCH_COEF10	Right channel Elliptic4_HPF_s1_b_R (2)	24
0x0530	DAC0_NCH_COEF11	Right channel Elliptic4_HPF_s1_b_R (3)	24
0x0534	DAC0_NCH_COEF12	Left channel Elliptic4_HPF_s2_gain_L	24
0x0538	DAC0_NCH_COEF13	Left channel Elliptic4_HPF_s2_b_L(1)	24
0x053C	DAC0_NCH_COEF14	Left channel Elliptic4_HPF_s2_b_L (2)	24
0x0540	DAC0_NCH_COEF15	Left channel Elliptic4_HPF_s2_b_L (3)	24
0x0544	DAC0_NCH_COEF16	Left channel Elliptic4_HPF_s2_b_L (2)	24
0x0548	DAC0_NCH_COEF17	Left channel Elliptic4_HPF_s2_b_L (3)	24
0x054C	DAC0_NCH_COEF18	Right channel Elliptic4_HPF_s2_gain_R	24
0x0550	DAC0_NCH_COEF19	Right channel Elliptic4_HPF_s2_b_R(1)	24
0x0554	DAC0_NCH_COEF20	Right channel Elliptic4_HPF_s2_b_R (2)	24
0x0558	DAC0_NCH_COEF21	Right channel Elliptic4_HPF_s2_b_R (3)	24
0x055C	DAC0_NCH_COEF22	Right channel Elliptic4_HPF_s2_b_R (2)	24
0x0560	DAC0_NCH_COEF23	Right channel Elliptic4_HPF_s2_b_R (3)	24
0x0564	DAC0_NCH_COEF24	Left channel Elliptic4_HPF_s3_gain_L	24
0x0568	DAC0_NCH_COEF25	Left channel Elliptic4_HPF_s3_b_L(1)	24
0x056C	DAC0_NCH_COEF26	Left channel Elliptic4_HPF_s3_b_L (2)	24

Offset address	Name	Description	Width
0x0570	DAC0_NCH_COEF27	Left channel Elliptic4_HPF_s3_b_L (3)	24
0x0574	DAC0_NCH_COEF28	Left channel Elliptic4_HPF_s3_b_L (2)	24
0x0578	DAC0_NCH_COEF29	Left channel Elliptic4_HPF_s3_b_L (3)	24
0x057C	DAC0_NCH_COEF30	Right channel Elliptic4_HPF_s3_gain_R	24
0x0580	DAC0_NCH_COEF31	Right channel Elliptic4_HPF_s3_b_R(1)	24
0x0584	DAC0_NCH_COEF32	Right channel Elliptic4_HPF_s3_b_R (2)	24
0x0588	DAC0_NCH_COEF33	Right channel Elliptic4_HPF_s3_b_R (3)	24
0x058C	DAC0_NCH_COEF34	Right channel Elliptic4_HPF_s3_b_R (2)	24
0x0590	DAC0_NCH_COEF35	Right channel Elliptic4_HPF_s3_b_R (3)	24

9.5.5.5.7 ST_DRC_COEF memory mapping

Offset address	Name	Description	Width
0x0D88	ST_DRC_RISE	ST DRC rise coef	16
0x0D8C	ST_DRC_FALL	ST DRC fall coef	16
0x0D90	ST_DRC_LMT	ST SRC LMT coef	16
0x0D94	ST_DRC_THD	ST DRC THD coef	16
0x0D98	ST_DRC_RATIO	ST DRC ratio coef	16
0x0D9C	ST_DRC_CG_VAR	ST DRC CG variate coef	16
0x0DA0	ST_DRC_ATK_RATE	ST DRC ATK rate coef	16
0x0DA4	ST_DRC_ATK_RATE_EX	ST DRC ATK rate external coef	16
0x0DA8	ST_DRC_HOLD	ST DRC hold coef	16
0x0DAC	ST_DRC_RLS_RATE	ST DRC RLS rate coef	16
0x0DB0	ST_DRC_RLS_RATE_EX	ST DRC RLS rate external coef	16
0x0DB4	ST_DRC_EXTEND_LMT	ST DRC extend LMT coef	16
0x0DB8	ST_DRC_EXTEND_THD	ST DRC extend THD coef	16
0x0DBC	ST_DRC_EXTEND_RATIO	ST DRC extend ratio coef	16
0x0DC0	ST_DRC_EXTEND_HOLD	ST DRC extend hold coef	16
0x0DC4	ST_DRC_EXTEND_LEVEL_L	Reserved	16
0x0DC8	ST_DRC_EXTEND_LEVEL_H	Reserved	16

9.5.5.6 FIFO memory mapping

9.5.5.6.1 DSP FIFO memory mapping

DSP FIFO	
Name	Offset address
DAC0 FIFO0	0x0
DAC0 FIFO1	0x4
DAC1 FIFO0	0x8
DAC1 FIFO1	0xC
ADC0 FIFO0	0x10
ADC0 FIFO1	0x14
ADC1 FIFO0	0x18
ADC1 FIFO1	0x1C
ADC2 FIFO0	0x20
ADC2 FIFO1	0x24
ADC3 FIFO0	0x28
ADC3 FIFO1	0x2C
AUDPLY DSP FIFO0(from AP)	0x30
AUDPLY DSP FIFO1(from AP)	0x34
AUDRCD DSP FIFO0(to AP)	0x38
AUDRCD DSP FIFO1(to AP)	0x3C

9.5.5.6.2 OFLD FIFO memory mapping

AP FIFO	
Name	Offset address
AUDPLY AP FIFO0	0x0
AUDPLY AP FIFO1	0x4
AUDPLY AP FIFO2	0x10
AUDPLY AP FIFO3	0x14
AUDRCD AP FIFO0	0x8
AUDRCD AP FIFO1	0xC
AUDRCD AP FIFO2	0x18
AUDRCD AP FIFO3	0x1C

9.6 SRC_48K

9.6.1 Overview

This module is used to covert the sample rate for assisting the music play. It can work in normal mode and high speed mode that is decided by software. If software read data from dsp_src_fifo by normal rate 48k, then software writes the input data into src normally and src work normally. If software want high speed, software should write input data and read data by high speed, then src work with high speed. As a result, software can get data from rxfifo as much as possible.

9.6.2 Features

- Support 8k/9.6k/11.025k/12k/16k/22.05k/24k/32k/44.1k/48k input sample rate.
- Support change input sample rate when the module is idle.
- Support pause and restart SRC
- Support enable and disable SRC
- Limiter is 24 bit
- Support bypass interpolation filter
- Support 20-bit width symmetrical coefficient, not configurable
- Support two channel
- TX FIFO RAM size is 512×48 bits
- RX FIFO RAM size is 512×48 bits
- Support two ports RAM

9.6.3 Function Description

The dsp src top module consists of dsp src regfile, dac src, rxfifo ctrl, txfifo ctrl. It receives data and covert the sample rate.

The regfile is an APB slave, which is used to configure the dsp src top module. Therefore, correct configuration should be sent to the regfile before starting the module.

The rx_fifo is used to store the data, which will be processed by the src module. The rx_fifo_ctrl module controls rx_fifo and generates related interrupts & states.

When the rx_fifo has data and the dac_src module is enabled, the module will process the data and generate new data, whose sample rate is 48K. There is a ram in dac_src module, used to store medium data. Which is generated during processing. It will affect the output. Therefore, src_clr command must be sent before restart the module everytime.

The tx_fifo is used to store the data, which has been processed by the src module. The tx_fifo_ctrl module controls tx_fifo and generates related interrupts & states.

The following table show the cycles for one input data converting.

Table 9-12 cycles for one input data

Up sample	cycles
8k	264
9.6k	1

Up sample	cycles
11.025k	210
12k	172
16k	129
22.05k	210
24k	86
32k	129
44.1k	210
48k	1

Table 9-13 cycles for one output data

Up sample	Cycles/output	Speed(120M)
8k/12k/16k/24k	43	120M/43/48k=58
9.6k/48k	1	
11.025k	70	120M/70/48k=35.7
22.05k	150	120M/150/48k=16.6
32k	86	120M/86/48k=29
44.1k	325	120M/325/48k=7.68

9.6.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.6.5 Application Notes

When software wants to start a SRC process, follow steps as below:

Step 1 Sent src_clr command to clear the ram in dac_src module;

Step 2 Configure the regfile, such as src_mode, fifo_thld and interrupt enable;

Step 3 Enable the dac_src module;

Step 4 Send audio data into the RX_FIFO, this IP support two channel, if there is only one channel data, please sent it into the right channel;

Step 5 Read 48k data from TX_FIFO;

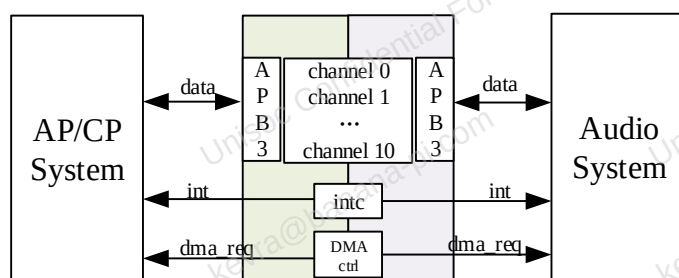
End

9.7 MCDT

9.7.1 Overview

MCDT module is designed as a digital interface, for sound stream transmission between audio System and other AP or CP system.

Figure 9-28 MCDT connect relation



There many data path between Audio system and AP/CP system for MP3 play or record, Voice or Phone:

- AP phone/voice play and record
- MP3 play and record
- DSP MP3 decode for AP
- TWG phone
- GSM phone
- VoLTE
- CP V_DL
- CP V_UL
- AP V_DL
- AP V_UL

9.7.2 Features

- Two APB3 slave interface, one for Audio processor, another for AP/CP system access
- 11 channels for DAC path and 9 for ADC path
- Each channel FIFO has independent watermark for empty and full control.
- Single port RAM: 10240 × 32 bit

9.7.3 Function Description

9.7.3.1 APB Interface

- Support APB3 protocol
- APB bus0 for Audio system access
- APB bus1 for AP/CP system access

9.7.3.2 DMA Interface

- dma req 0,independent enable bit for each dma request

TX dma request 0	RX dma request 0
dma_ch0_tx_req0	dma_ch0_rx_req0
dma_ch1_tx_req0	dma_ch1_rx_req0
dma_ch2_tx_req0	dma_ch2_rx_req0
dma_ch3_tx_req0	dma_ch3_rx_req0
dma_ch4_tx_req0	dma_ch4_rx_req0
dma_ch5_tx_req0	dma_ch5_rx_req0
dma_ch6_tx_req0	dma_ch6_rx_req0
dma_ch7_tx_req0	dma_ch7_rx_req0
dma_ch8_tx_req0	dma_ch8_rx_req0
dma_ch9_tx_req0	dma_ch9_rx_req0

- dma req 1 for AP/CP processor,independent enable bit for each dma request

TX dma request 1	RX dma request 1
dma_ch0_tx_req1	dma_ch0_rx_req1
dma_ch1_tx_req1	dma_ch1_rx_req1
dma_ch2_tx_req1	dma_ch2_rx_req1
dma_ch3_tx_req1	dma_ch3_rx_req1
dma_ch4_tx_req1	dma_ch4_rx_req1
dma_ch5_tx_req1	dma_ch5_rx_req1
dma_ch6_tx_req1	dma_ch6_rx_req1
dma_ch7_tx_req1	dma_ch7_rx_req1
dma_ch8_tx_req1	dma_ch8_rx_req1
dma_ch9_tx_req1	dma_ch9_rx_req1

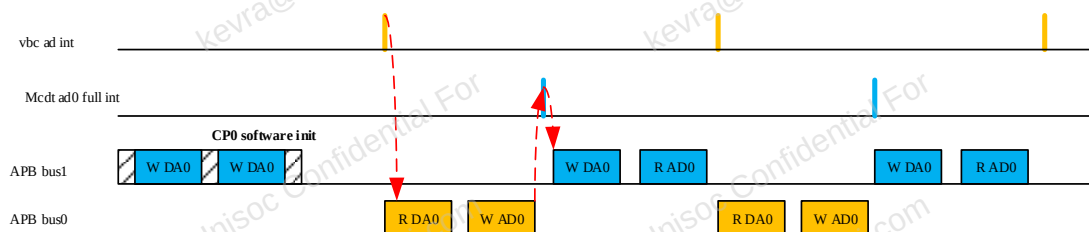
9.7.3.3 Interrupt Interface

- mcdt_int1_aud for Audio processor
- mcdt_int0_ap for AP system
- mcdt_int0_cp0 for CP0 system
- mcdt_int0_cp1 for CP1 system
- mcdt_int0_cp2 reserved

9.7.3.4 FIFO Control

DA0 and AD0 FIFO for CP0 TD-LTE/WCDMA channel, 8kHz sample rate, DA0 and AD0 FIFO full /empty watermark set to 40

Figure 9-29 FIFO control flow 1



Step 1 CP0 initial DA FIFO, writes 80 in DA FIFO

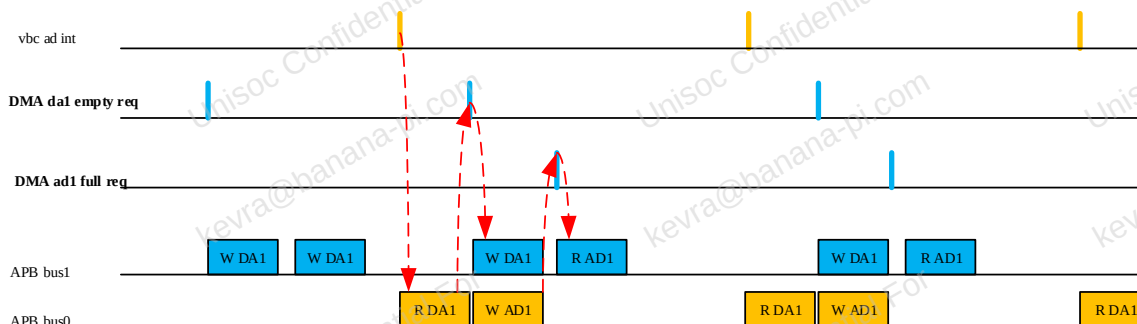
Step 2 When Audio DSP receives the VBC AD interrupt, Audio DSP reads 40 data from DA FIFO, then write 40 data into AD FIFO, MCDT trigger an AD FIFO full interrupt

Step 3 After CP0 receives the MCDT AD FIFO full interrupt, CP0 write 40 data into DA FIFO, then reads 40 data from AD FIFO

Step 4 Repeat step2 and step3 until data transfer end.

End

DA1 and AD1 FIFO for AP MP3 play and record channel, 48kHz sample rate, DA and AD FIFO full /empty watermark set to 240



Step 1 AP initial DMA, DA and AD FIFO full/empty watermark set to 240, and enable MCDT MP3 play and record channel dma_enable, then DA FIFO trigger a DMA request to DMA controller, DMA write 480 data into DA FIFO

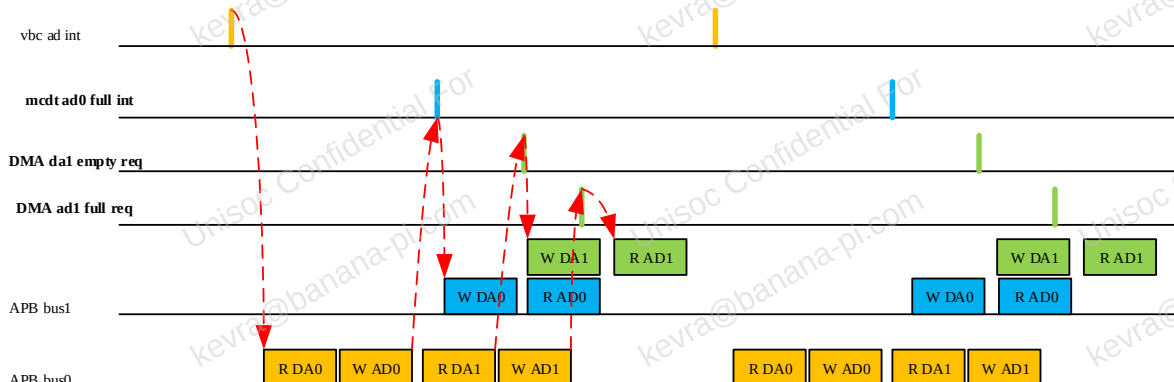
Step 2 when Audio DSP receives the VBC AD interrupt, Audio DSP reads 240 data from DA FIFO, DA FIFO trigger a DMA request to DMA controller, Audio DSP write 240 data into AD FIFO, AD FIFO trigger a DMA request to DMA controller

Step 3 when DMA receives MCDT DA FIFO DMA request, DMA write 240 data into DA FIFO, when DMA receives MCDT AD FIFO DMA request, DMA read 240 data from AD FIFO

Step 4 repeat step2 and step3 until data transfer end.

End

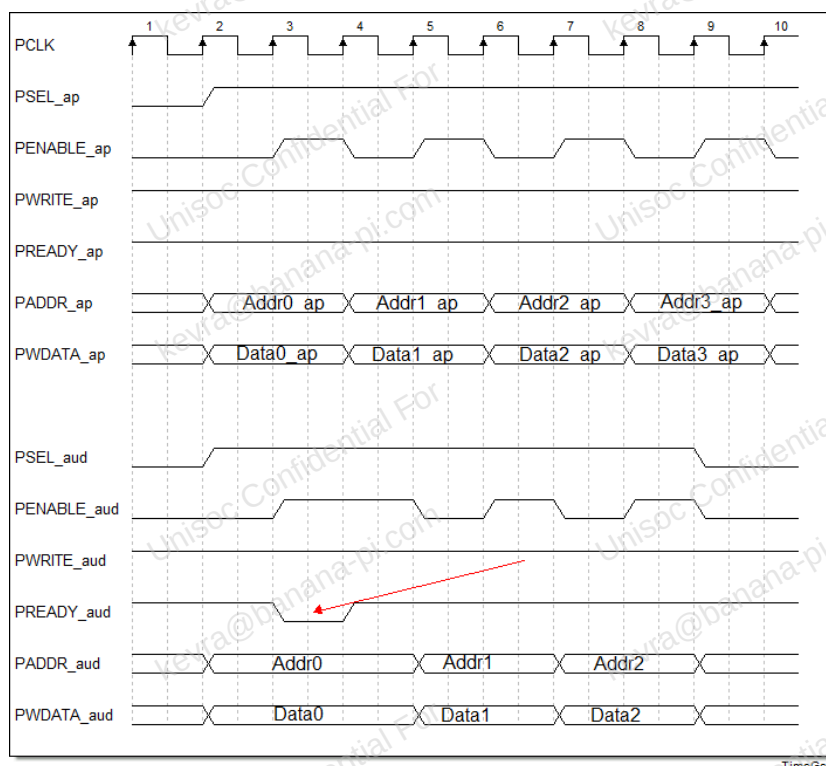
DA0 and AD0 FIFO for CP0 TD-LTE/WCDMA channel, 8kHz sample rate, DA0 and AD0 FIFO full /empty watermark set to 40. DA1 and AD1 FIFO for AP MP3 play and record channel, 48kHz sample rate, DA and AD FIFO full /empty watermark set to 240



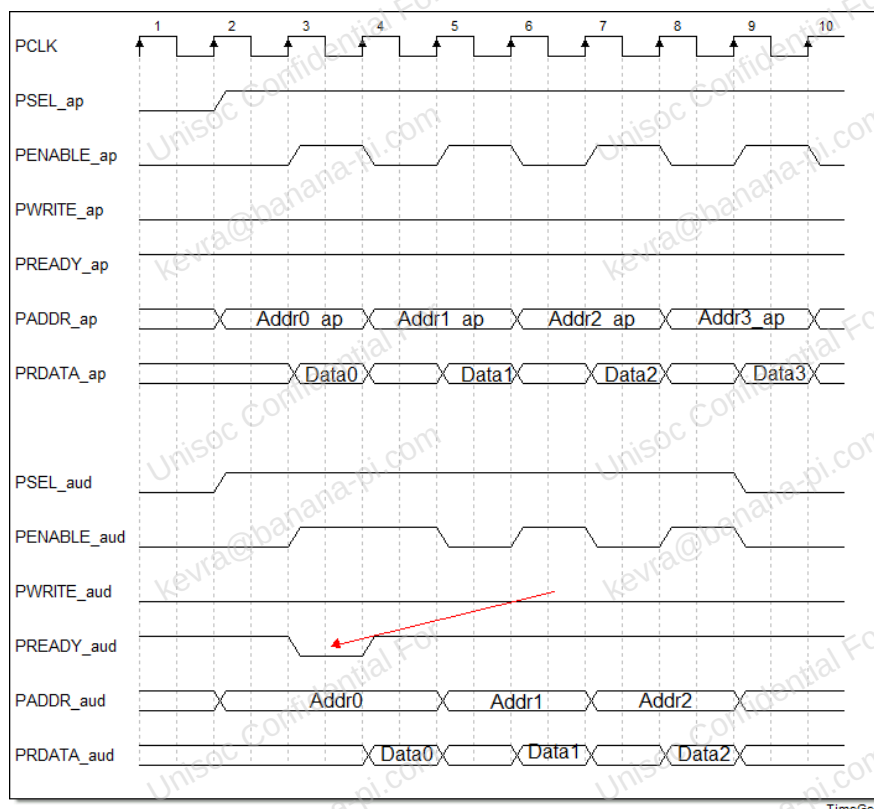
9.7.3.5 BUS Access control

APB bus0 and APB bus1 access ram conflict resolution

- Write:



- Read:



9.7.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.7.5 Application Notes

9.7.5.1 DMA channel control

MCDT has 11 channels for dac path, and can select which channel to use by configuring dma channel select register, including MCDT_DMA_REQ_CFG0-MCDT_DMA_REQ_CFG3 and MCDT_DMA_ACK_CFG0-MCDT_DMA_ACK_CFG3.

It should be noted that when selecting channel, you should configure channel request register and channel ack register both, or dma request and ack cannot handshake successfully. For example, when you configure mcdt_dac_dma_ch0_req_sel 0, then the mcdt_dac_dma_ch0_ack_sel should be configured 0, which means you use dac0 channel. Similarly, if you configure mcdt_dac_dma_ch0_req_sel 3, then mcdt_dac_dma_ch3_ack_sel should be configured 0 (because you use dac0 port send dac3 channel request, and ack is accepted by dac0 port, you should select the ack to channel3, so that handshake successfully). Meanwhile, the channel which you selected MCDT_DMA_EN should not be forgotten to configure.

And 9 channels for adc path is the same.

9.7.5.2 Application scenario description

The value of setting MCDT_DMA_REQ_CFG and MCDT_DMA_ACK_CFG registers: DMA physical channel is send to the other subsystems, is serving for AP and CP. For example, AP can support 7 channels data

transmission at the same time, you should not limit on fixed 7 channels. And the two group registers are providing the channel selectivity.

9.8 PDM

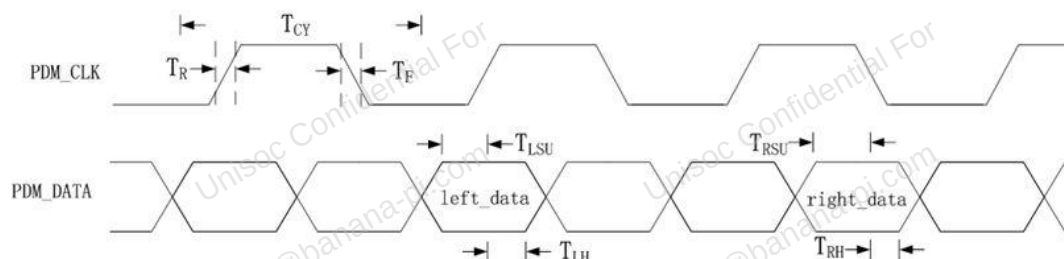
9.8.1 Overview

PDM (pulse density modulation) is audio interface. This module is the digital part for processing the input of dmics. The audio data out can be transmitting by IIS interface or can be accessed by dma request or interrupt. All external dmics must be the same. the pdm data are send to vbc and vad. The data in vad is used for voice trigger. The data in vbc is used for acoustic echo cancel.

9.8.2 Features

- Support 20M, 26M, 38.4M and 6M. the clock of pdm is 6M when in low power mode.
- Support dma access and CPU interrupt access
- 128 × 27 asynchronous FIFO
- Support programmable Fs for ADC: 8/16/24/32/48 kHz. In low power mode only supports 16k.
- Support IIS master interfaces, where only IIS0 support compressed mode
- Support both left and right ADC channel for stereo MIC
- Support dual multiplexed channels

Figure 9-30 DMIC interface timing diagram



9.8.3 Function Description

The input dmics data is processed by 3 decimation filters. The first decimation needs at least 3 cycles for calculation. The second has 25 coefs and down sample is 5. So the second decimation needs at least 10 cycles; the third decimation has 165 coef and down sample rate is 5, so the third decimation needs at least 33 cycles for calculation.

For the second decimation filter, the input sample rate is 400k(16k), 1200k(48k), 600k(24k), 800k(32k), 200k(8k) Hz.

For 20m/26m/38.4m, all cases have enough cycles.

In low power mode, the clock for pdm is 6MHz. the typical application is voice trigger. Only 16kHz are supported.

9.8.3.1 sample rate16k

The dmic clock is divided by clock of pdm (such as 6/20/26/38.4MHz)

The divider = xtl clock/dmic_clk

xtl clock	div→Mclock	div→Mclock	div→Mclock	div→Mclock	div→Mclock	div→Mclock
6M	15→400K	12→500K	10→600K	6→1000K	5→1200K	3→2000K
26M	52→500K	26→1000K	13→2000K			
38.4M	64→600K	48→800K	32→1200K	24→1600k	16→2400k	
20M	50→400K	25→800K	20→1000K	10→2000k		

9.8.3.2 sample rate48k

900K/1800K/3600K/7200K can't be supported.

1500K/3000K/6000K can't be supported,

xtl clock	div→Mclock	div→Mclock	div→Mclock	div→Mclock	div→Mclock	div→Mclock
26M		26→1000K	13→2000K			
38.4M	64→600K	48→800K	32→1200K	24→1600k	16→2400k	16→4800k
20M	50→400K	25→800K	20→2000k	10→2000k		

9.8.3.3 sample rate32k

xtl clock	div→Mclock	div→Mclock	div→Mclock	div→Mclock	div→Mclock	div→Mclock	div→Mclock
26M		26→1000K	13→2000K				
38.4M	64→600K	48→800K	32→1200K	16→2400k	12→3200k	8→4800k	24→1600k
20M	50→400K	25→800K	10→2000K	5→4000k			

9.8.3.4 sample rate24k

450K/900K/1800K/3600K/ can't be supported.

750K/1500K/3000K can't be supported,38.4/3=12.8

xtl clock	div→Mclk	div→Mclk	div→Mclk	div→Mclk	div→Mclk	div→Mclk	div→Mclk	div→Mclk
26M	52→500K	26→1000K	130→200K	75→400K				
38.4M	64→600K	48→800K	32→1200K	24→1600k	16→2400k	128→300k	96→400k	192→200k
20M	100→200K	50→400K	25→800K	20→1000K				

9.8.3.5 sample rate8k

xtl clock	div→Mclk	div→Mclk	div→Mclk	div→Mclk	div→Mclk	div→Mclk	div→Mclk	div→Mclk
26M	104→250K	52→500K	26→1000K					
38.4M	128→300K	96→400K	64→600K	48→800K	32→1200K	192→200K		
20M	100→200K	50→400K	25→800K	20→1000K	40→500K	80→500K		
6M	30→200K	24→250K	20→300K	15→400K	12→500K	10→600K	6→1000K	5→1200K

The input data number (in_num) and the output number(out_num) has relationship as below:

$$\text{Out_num} = \text{in_num} \times (\text{rf_adc_decim_1st_up_num}) / \text{rf_adc_decim_1st_down_num} / 5 / 5$$

For example:

When the audio sample rate is 24k, the dmick clock is 1200k, the rf_adc_decim_1st_up_num is 1, the rf_adc_decim_1st_down_num is 2. the input number is 50050, the output num is $50050 \times 1 / 2 / 5 / 5 = 1001$.

9.8.3.6 IIS module

There are three IIS modules, namely iis0, iis1 and iis2. The iis0 can output data of ADC0 or output data of ADC0 and ADC1; the iis1 can output data of ADC1; the iis2 can output data of ADC2. Each of the above ADC has 2 mics. Among the above IIS modules, only the iis0 supports compressed mode.

The following describes the IIS normal and compressed mode.

If there are 2 mics, IIS0 interface can only be normal, in this case, no compressed mode.

Figure 9-31 PDM IIS0 DATA

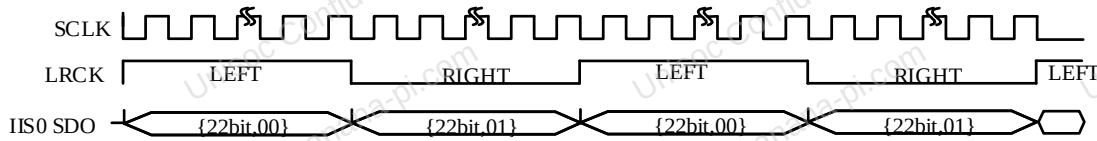


Figure 9-32 PDM IIS1 DATA

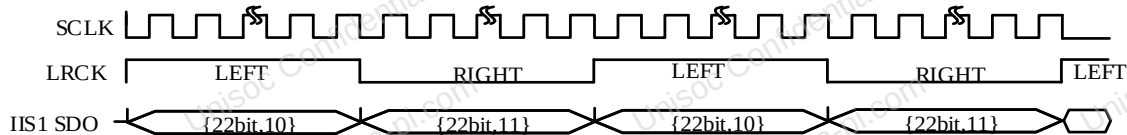
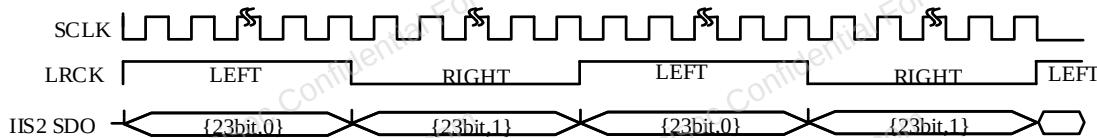
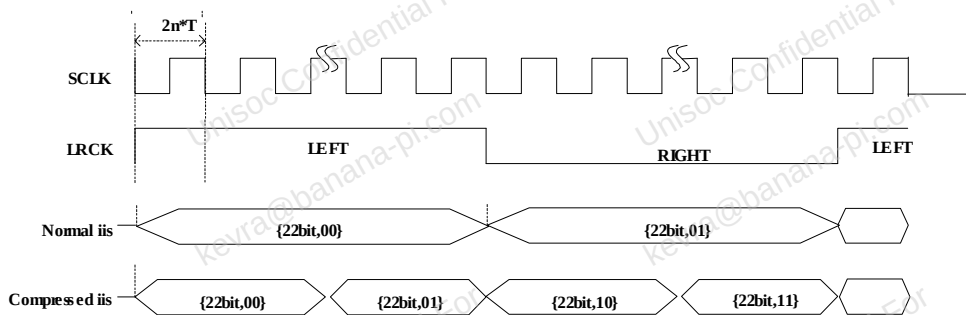


Figure 9-33 PDM IIS2 DATA



If there are 4 mics, IIS0 interface is compressed IIS format. The bit [1:0] is used to identify the data sequence.

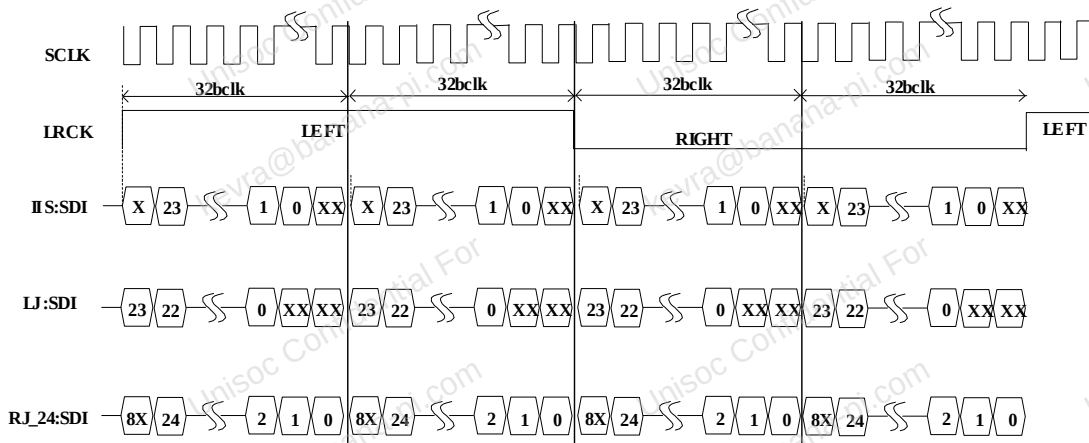
Figure 9-34 PDM IIS DATA



Compressed IIS format means 4 valid data in one lrck cycles.

The compressed IIS supports 3 data format which is IIS compatible, left justified, right justified.

Figure 9-35 pdm data 24bit slot length32bit



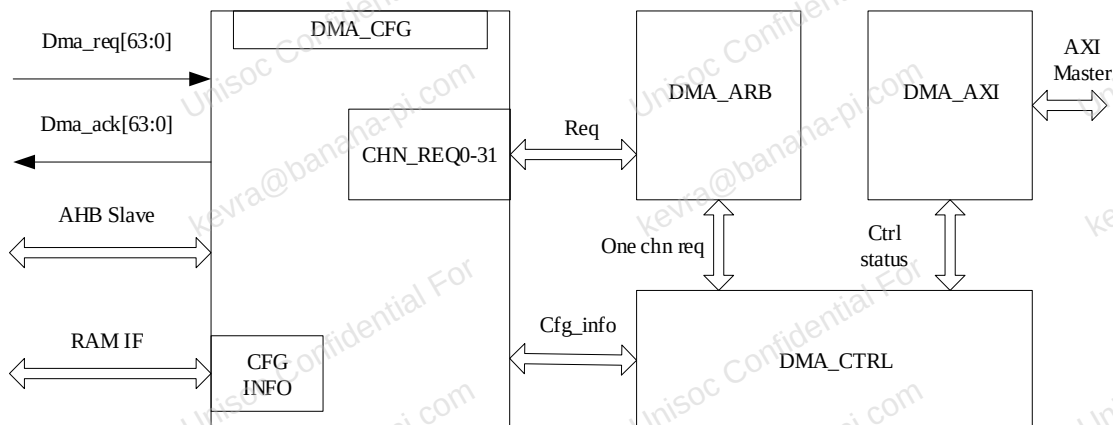
9.8.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.9 DMA

9.9.1 Overview

Figure 9-36 DMA data path



9.9.2 Features

- Arm® AMBA® 3 AXI Specification.
- Two 64-bit AXI masters for read and write operation.
- One AHB 32-bit slave for configuration only supports word access to AHB slave port.
- Support Byte/Half-word/word transfer on AXI Read/Write Master.
- Not support Arm® AMBA® EXOKAY/SLVERR/DECERR response.

- Support Arm® AMBA® INCR transaction, burst length is calculated by transaction Byte length.
- Support max outstanding number is 16.
- Support 64GB address space.
- AXI Master Big-endian or Little-endian.
- Support 32 channels assigned to hardware request by Channel ID;
- One physical channel shared by 32 Logic channels.
- 4-level configurable priority. 3 has the highest and 0 has the lowest priority. For the channels with the same priority, round-robin would be preferred.
- A-synchronized/Synchronized hand-shake request & acknowledge.
- Accept request from both hardware and software.
- Un-aligned word accesses, for both read and write operation.
- Byte/Half-word switch scheme.
- Support link list for scatter and gather transfer.
- Support one channel to start the other channel.
- Configurable fragment wait time between two successive fragments.
- One request for fragment/block/transaction/link-list.
- Automatically disable channel after requests done.
- Support 128K-1 Byte fragment length.
- Support 128K-1 Byte block length (Block length may be less than fragment length in Standard Channel, while forbidden in Full channel).
- Support 256M-1 Byte transaction length.
- Support transfer step, ranged from – (32K-1) Bytes to (32K-1) Bytes, only for full DMA channel.
- Support fragment step, ranged from – (32K-1) Bytes to (32K-1) Bytes, only for full DMA channel.
- Support block step, ranged from – (128M-1) Bytes to (128M-1) Bytes, only for full DMA channel.
- Address-fix transfer for either Source side or destination side.
- Address-wrap transfer for either Source side or destination side.
- Pause DMA or pause logical channel.
- Interrupt of fragment, block, transaction and link list, with raw and masked status.
- Interrupt of Error configuration, and support errors status.
- Support hardware request to be re-allocated by Channel ID.
- Support automatically sleep when all requests served, and wakeup when hardware request active.
- Support software retention if power domain of DMA shut down.

9.9.3 Function Description

9.9.3.1 General Description

DMA Controller support fast data transfers to off-load the processor. With DMA, specific devices on AXI and APB can complete quick data movement from/to memories, internal RAM or External SRAM. DMA Controller can also be used to connect any two devices other than memories, as long as they can be addressed in memory space.

9.9.3.2 Standard & Full DMA Channels

There are two types of DMA channels. One is called “Standard” DMA channel, and the other is “Full” DMA channel.

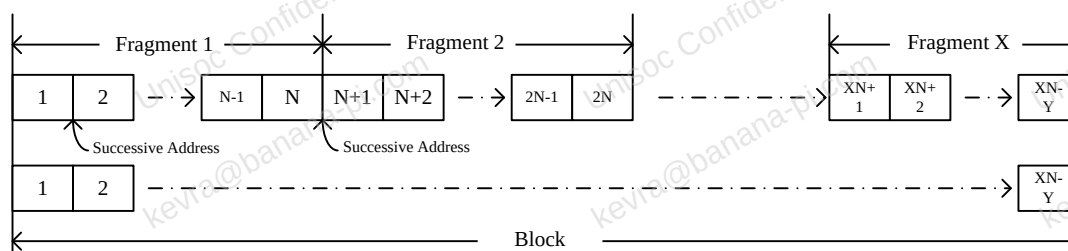
For most applications, a standard DMA channel would be sufficient, with least configurations and simplest control flow. AXI sequential transfer and FIFO transfer would be efficiently managed, with Byte, half-word or word size.

For special applications, such as image rotation or complex data arrangement, Full DMA channel support address wrap transfer, configurable transfer step and link list mode.

9.9.3.3 Data Structure

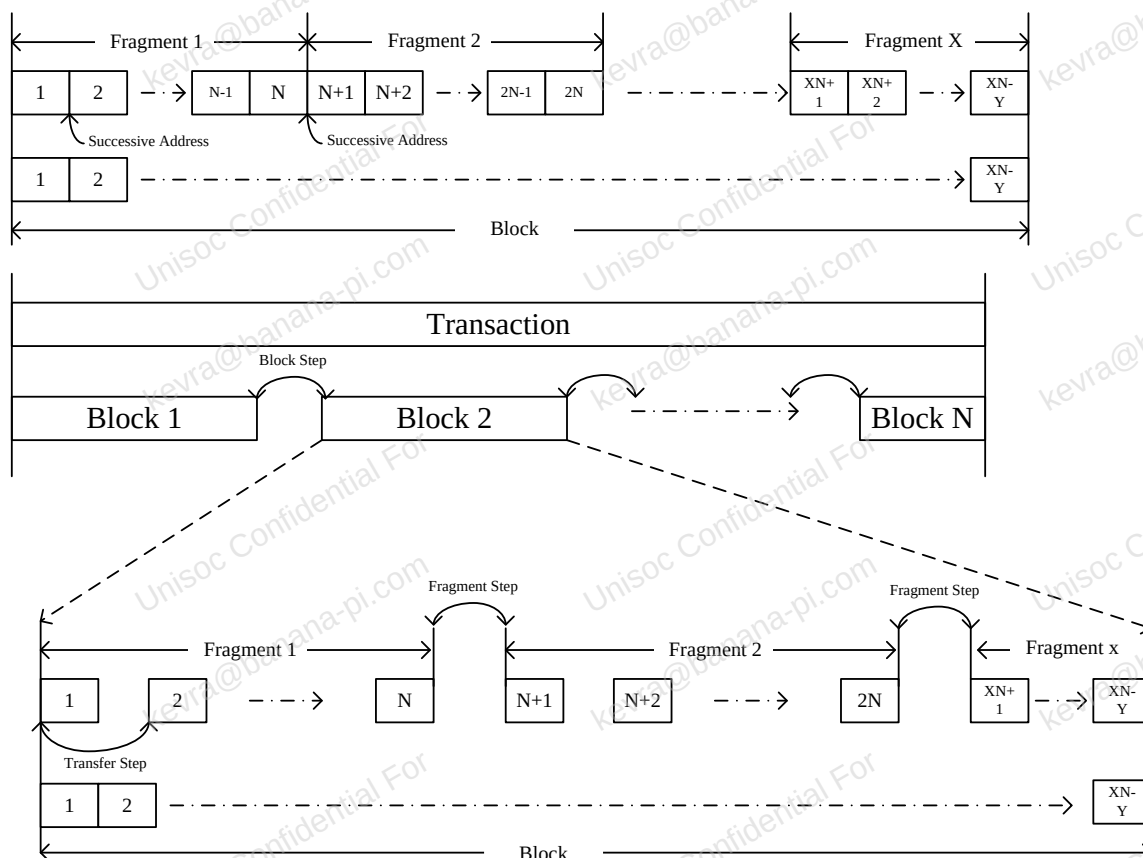
- For Standard DMA channel

Figure 9-37 DMA data structure for standard channel



- For Full DMA channel

Figure 9-38 DMA data structure for full channel



9.9.3.4 DMA Pause

There are two levels of pause action implemented in DMA controller.

One is “controller pause” suitable for the whole DMA controller. Once the controller pause occurred, the DMA controller would stop all the channels after the current fragment/block completed, hold the accepted and subsequently requests. DMA would not sleep during pause state.

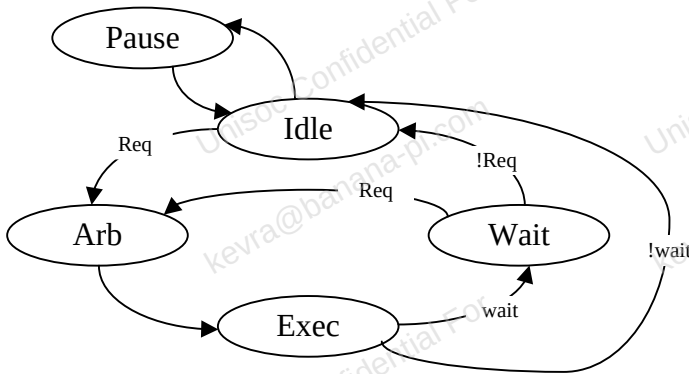
The other is “channel pause” suitable for the single DMA channel. When the channel pause occurred, the channel would be temporarily hold and stop all the transfer action.

9.9.3.5 Channel Priority and Arbitration

Each channel may be configured to be one of total 4 level priorities, to effective manage the bandwidth among all logical channels. If there is more than one request occurring, DMA controller would firstly judge their priorities and then select the highest one to serve.

For the channels having the same level priority, DMA controller would serve them by Round-robin way.

Figure 9-39 DMA arbitration state machine



9.9.3.6 Address Fix Mode

Both Standard and Full DMA channel support address-fix transfer, on either source or destination side. This mode is optimized for UART/SPI etc peripherals, and fragment length “FRAG_LEN” would be configured equal to the FIFO watermark.

9.9.3.7 Address Wrap Mode

Full-DMA channel support data movement of ring-buffer and double-buffer, as illustrated below. Once the address reaches the “WPPT”, the next address jumps to the “WPTO” address after completing the “WPPT” data transfer.

Attention the data at “WPPT” isn’t moved to the destination address.

Figure 9-40 DMA data movement example-1

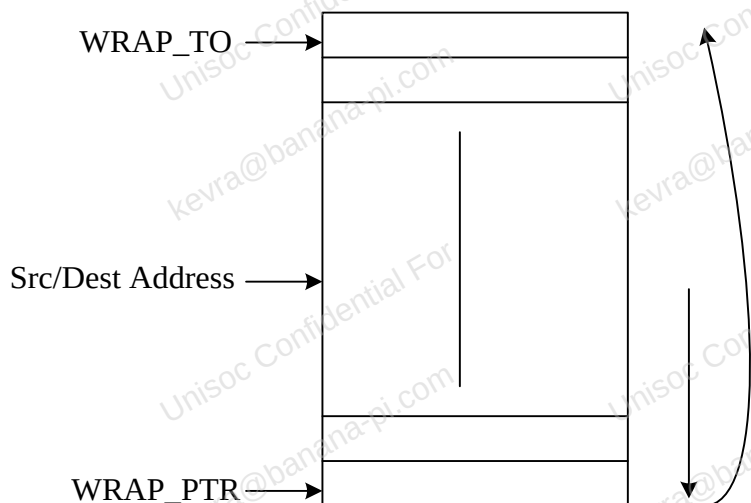
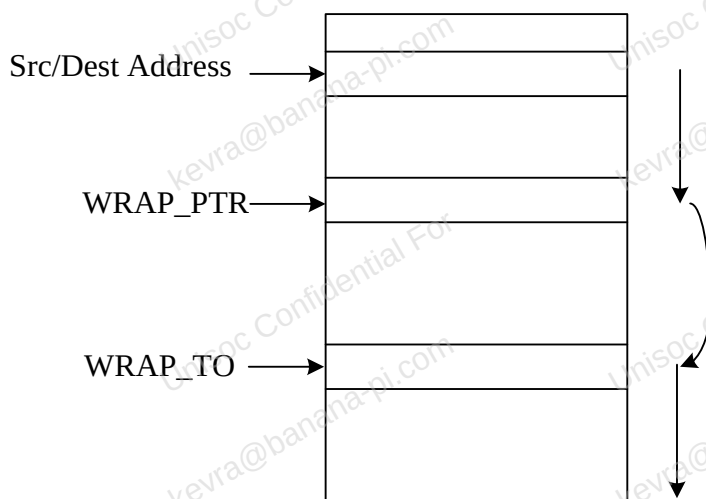


Figure 9-41 DMA data movement example-2



9.9.3.8 Endian & Data Switch

DMA controller would be configured to be big-endian or little-endian on AXI master ports. The detailed information on AXI big-endian and little-endian would be preferred to AMBA® 3 Specifications.

Beside AXI master endian configuration, each channel would separately switch data as scheme below:

9.9.3.9 Un-aligned Word Access

Word access on AXI bus support burst type of address un-aligned to word boundary. If DMA was configured to read/write un-aligned address with word size, when DMA read, the DMA would start axi burst to read memory by the un-aligned start address, then the DMA discard redundancy Byte and selects valid data, when DMA write, the DMA would start un-aligned burst transfer, but the valid data information must be in the strobe signal.

this DMA AXI burst types are all INCR, the burst length is 1-16.

DMA probably had to read/write them by Bytes, as illustrated below.

Figure 9-42 DMA transfer examples

Start address:0x00
Transfer size:32bit
Burst type:INCR
Burst length:5 transfers

3	2	1	0	1st transfer
7	6	5	4	2nd transfer
11	10	9	8	3rd transfer
15	14	13	12	4th transfer
19	18	17	16	5th transfer

Start address:0x01
Transfer size:32bit
Burst type:INCR
Burst length:5 transfers

3	2	1	0	1st transfer
7	6	5	4	2nd transfer
11	10	9	8	3rd transfer
15	14	13	12	4th transfer
19	18	17	16	5th transfer

Start address:0x02
Transfer size:32bit
Burst type:INCR
Burst length:5 transfers

3	2	1	0	1st transfer
7	6	5	4	2nd transfer
11	10	9	8	3rd transfer
15	14	13	12	4th transfer
19	18	17	16	5th transfer

Start address:0x03
Transfer size:32bit
Burst type:INCR
Burst length:5 transfers

3	2	1	0	1st transfer
7	6	5	4	2nd transfer
11	10	9	8	3rd transfer
15	14	13	12	4th transfer
19	18	17	16	5th transfer

The red Byte is redundancy Byte.

For the memory space with un-aligned word access, Byte/Half-word/Word size access must be supported.

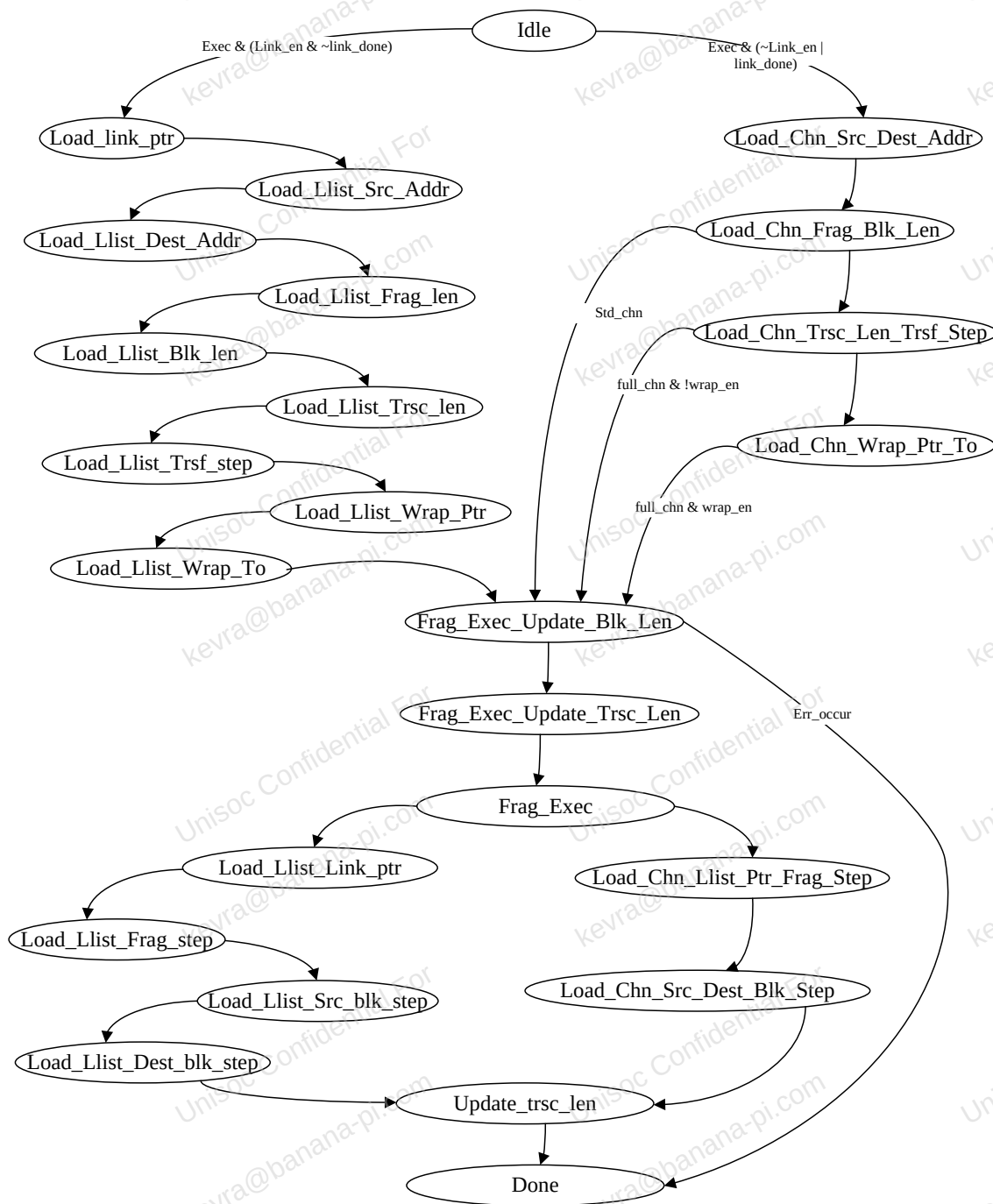
For Full DMA channel, if the step of transfer different with the data size (the address is not successive), un-aligned word access would turn to be total Byte access.

9.9.3.10 Word Packing

To maximum utilizing internal FIFO, and reduce AXI wait for un-aligned word access, Byte or half-word would be packed into word internal DMA.

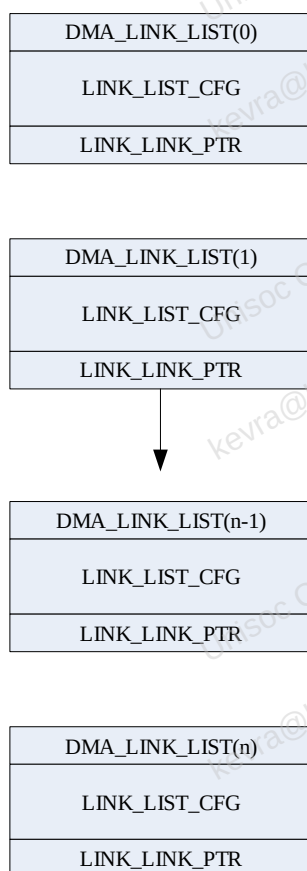
9.9.3.11 Channel Execution Flow

Figure 9-43 DMA channel execution flow



9.9.3.12 Link List

Figure 9-44 DMA link list



Software can create a few link list, and then connect these list, the link_list_end flag of the last list should be high means list sequence is end.

9.9.3.13 Hardware Request Re-allocated

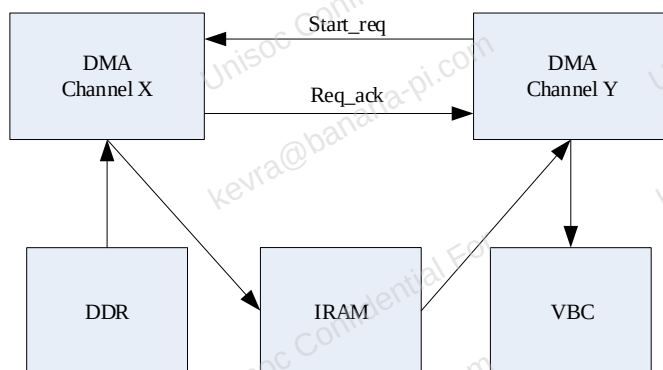
Every device (peripherals or accelerators) would be assigned to a hardware request, when DMA was implemented in SOC. DMA controller support each hardware request to be re-allocated to the Standard or Full DMA channels.

9.9.3.14 One Channel Start Other Channel

This Function is optimized for audio, firstly, the software must configure the only two channel registers, and ensure the two channels can't be used by any other hardware request or software request, and each channel can start the other channel.

DMA audio application as illustrated below.

Figure 9-45 DMA one channel start other channel

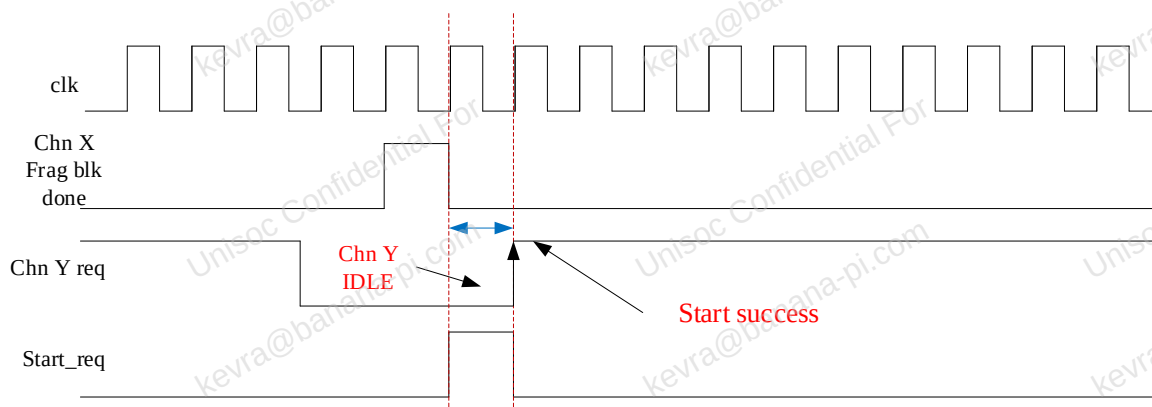


Firstly, CPU should start channel X only one time, when the IIRAM has enough data from channel X, CPU (or VBC) can start channel Y to shift data from IIRAM to VBC. when channel X is idle and a small amount data in IIRAM, channel Y can start channel X when channel Y frag_done or blk_done or trsc_done or llist_done.

Start control signal timing:

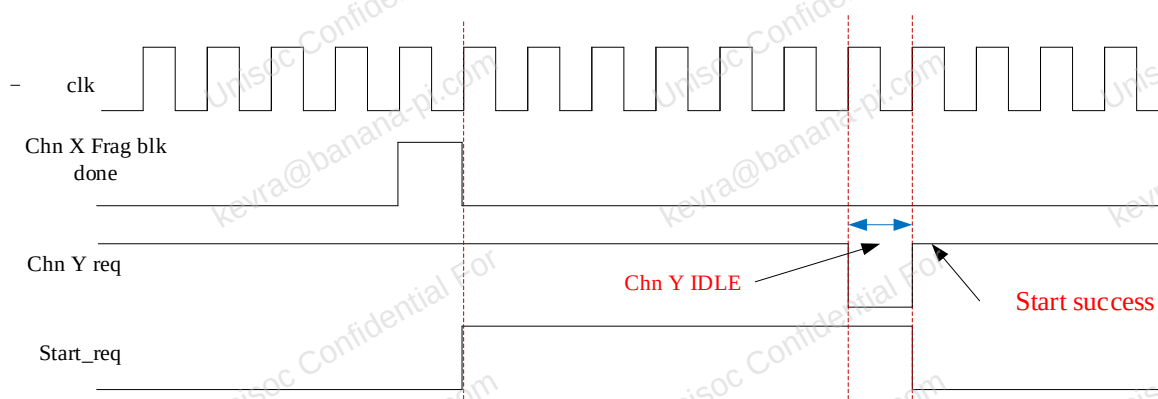
The channel Y is idle state when channel X start request is valid, then next clock cycle start channel Y is successful.

Figure 9-46 channel Y is idle while channel X start request



The channel Y is busy state when channel X start request is valid, so the channel X request will hold until channel Y is idle, when the channel Y is idle, then next clock cycle start channel Y is successful

Figure 9-47 channel Y is busy while channel X start request



9.9.3.15 Error Configuration

- Unaligned word/halfword access.
- Request for transaction or link list in Standard DMA channel.
- Zero Fragment/block/transaction length.
- Address wrap for Standard Channel.

9.9.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.9.5 Application Notes

- In Standard Channel, “frag_len” would be larger or smaller or equal to “blk_len”, and “blk_len” would be updated after each “frag_len” completed. “blk_cnt” would be no longer exists. “blk_len_rec_h/l” not available in Standard Channel.
- In Full Channel, “Frag_len”, “blk_len” and “trsc_len” would be any relationship of size. “blk_len” would not be updated after “frag_len” completed and “blk_len_rec_h/l” would be updated.
- In Standard Channel, not support 8GB address space
- In FULL Channel, support 64GB address space, but the llist_ptr can’t cross the 4GB boundary.

Dma_64b_top uses 64bits axi bus.

- When src/dst data size equal 2'b01,the all addr/step/length/wrap addr to/ptr should be half-word align.
- When src/dst data size equal 2'b10,the all addr/step/length/wrap addr to/ptr should be word align.
- When swt_mode !=2'b00, or data format is big-endian, the all address and length should be word align.

Link list pointer should be DWORD align.

9.10 IIS

9.10.1 Overview

IIS is a common digital audio interface specification. Base Band Chip IIS Interface can be used to implement a CODEC interface with external digital audio system working as host or slave. The IIS interface supports both IIS

and PCM data format. The interface can transmit and receive data simultaneously as well as transmit or receive data alternatively at a time. This document defines function and configuration of IIS block.

IIS module can be controlled both by Arm® processor or DSP.

9.10.2 Features

- Compliant with the IIS/PCM standard
- Support IIS, MSB-Justified and DSP data bus interface working on IIS mode
- Support Long Frame, Short Frame and Multi-Cycle Frame working on PCM mode
- Serial clock with programmable frequency
- Two 32-word-deep FIFOs minimizes processor overhead at high data rates
- LRCK polarity can be configured
- Wide selection of IIS data lengths 8,16,32bits
- Support master / slave mode
- Support MSB / LSB mode
- Transmit only/ receive only / transmit and receive mode
- Request 2 DMA
- Single interrupt line for multiple interrupt source events
- IIS and PCM master mode: support integer clock divider
- IIS master mode: support fraction clock divider – both variable and fixed serial clock mode
- PCM master mode: support fraction clock divider – only variable serial clock mode
- Support 32bit width of APB 2.0 configuration.
- Support multi data channels (parameter CHN_NUM controls only support 1 & 2)
- Support TX & RX FIFO depth configurable (parameters TX_FIFO_DEPTH & RX_FIFO_DEPTH only support 5/6/7/8 which means FIFO depth 32/64/128/256)
- Support RX only feature for master mode
- Support max 4 data channels
- Support IIS mode & slave mode tx_ng=1 to improve slave mode timing

9.10.3 Function Description

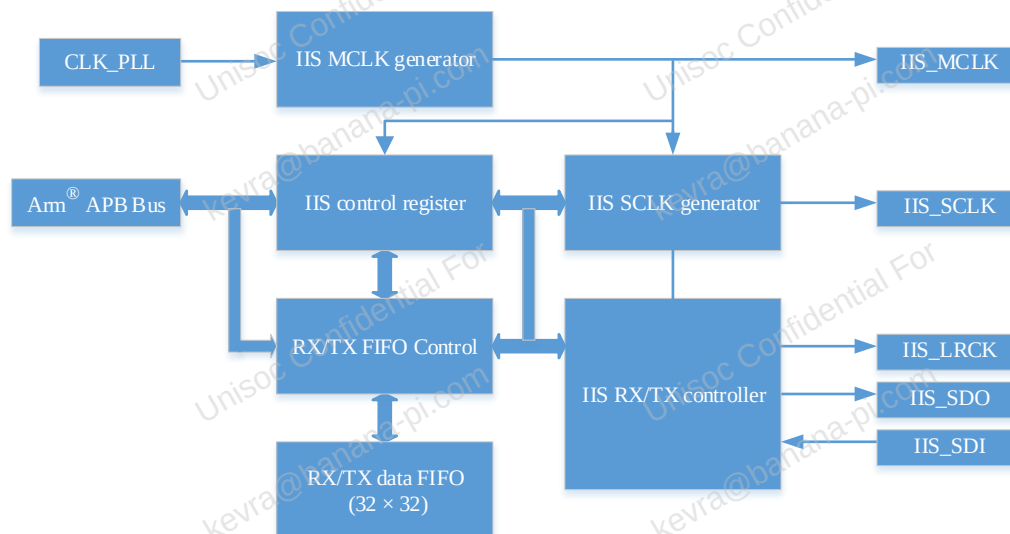
9.10.3.1 IIS interface

IIS module consists of RX/TX controller, Arm® processor APB interface control register, clock generator and FIFO control. Arm® program IIS register, enable IIS mode, starts RX/TX data transfer. Output data is written to TX FIFO and received data is stored in RX FIFO. Each FIFO is 32-bit width plus 32-address length.

MCLK generator generates main clock for IIS block. This clock also output as master clock during hand shaking. In slave mode, external clock source can be input, working as the IIS main clock. Clock divider value and control signal for MCLK come from global control register.

The IIS SCLK generator divides main clock according to a 16-bit clock divider register, which defines half SCLK clock width. It can be programmed from 0 to 32767.

Figure 9-48 IIS module diagram



IIS interface has 4 pins, IIS_SCLK is serial clock output, IIS_SDI is serial data input, IIS_SDO is serial data output. IIS_LRCK is left/right channel select. The device generating IIS_LRCK and IIS_SCLK is the master. IIS_LRCK and IIS_SCLK are output pins if working as master. They are input pins if working as slave.

Serial data is transmitted in two's complement format with the MSB first. Serial data can be sent at rising or falling edge of SCLK, at receiver, it must be latched at rising edge. When system word length is greater than the transmitter word length, the word is truncated (least significant data bits are set to '0') for data transmission. If the receiver is sent more bits than its word length, the bits after the LSB are ignored. On the other hand, if the receiver is sent fewer bits than its word length, the missing bits are set to zero internally.

Three bus formats are available in this module. Following Figures illustrate the relationship between the SCLK, LRCK and serial data I/O for different interface protocols. The polarity of LRCK is programmable. The bits per-channel is programmable up to 32bit.

IIS mode is where the MSB is available on the 2nd rising edge of SCLK following a LRCK transition.

Figure 9-49 IIS-compatible format

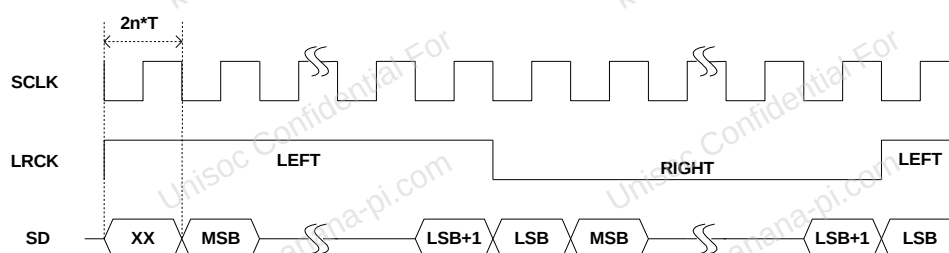
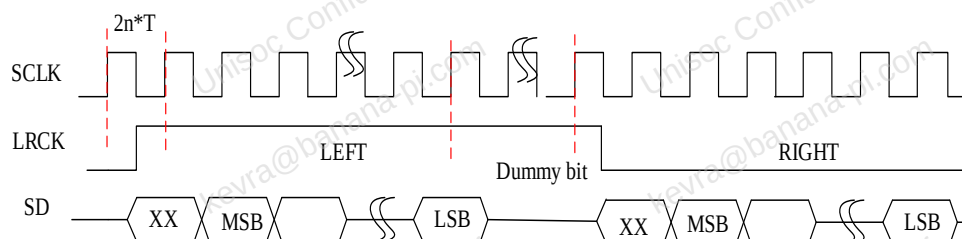


Figure 9-50 IIS-compatible format with dummy bit



MSB-justified mode is where the MSB is available on the first rising edge of SCLK following a LRCK transition.

Figure 9-51 MSB-justified format

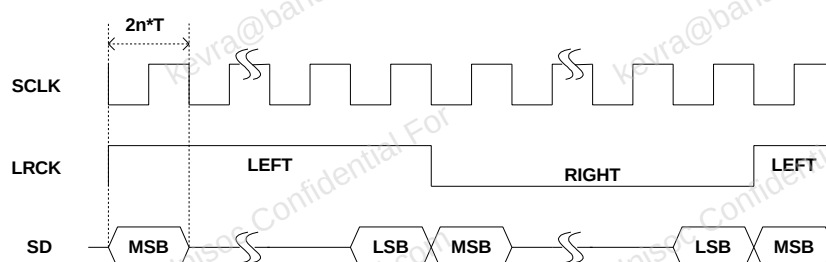
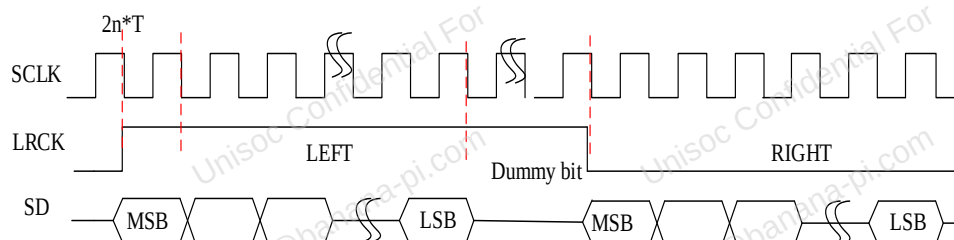
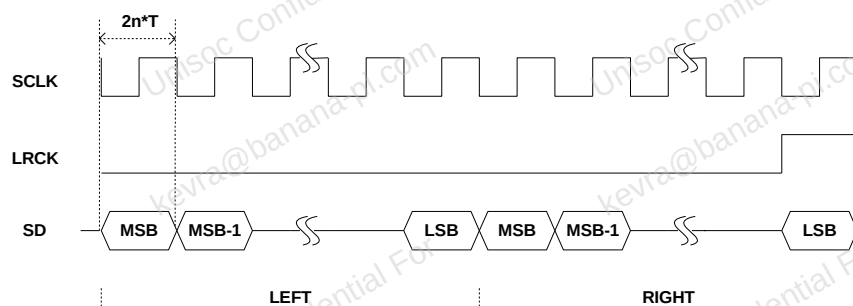


Figure 9-52 MSB-justified format with dummy bit



DSP mode is where the left channel MSB is available on either the 1st or 2nd rising edge of SCLK following a LRCK transition high. Right channel data immediately follows left channel data.

Figure 9-53 Synchronized format



NOTE

- T: 1/(system clock frequency)
- n: register programmable integer, n-1 = 0, ..., 32767

Table 9-14 IIS data format memory map

IIS_CTL bit1~0	Bit/channel	Data arrangement
00	8bit	Data on bit 31~24
01	16bit	Data on bit 31~15
1x	32bit	Data on bit 31~0

9.10.3.2 PCM interface

PCM interface is a direct voice interface connects to standard CODEC. The implementation is compliant with the MP-PCM requirements for voice transfer (8 kHz PCM_SYNC and 8 or 16 bits data). The four signals of the PCM interface are:

- PCM_CLK: PCM clock
- PCM_SYNC: PCM 8kHz synchronization signal
- PCM_OUT: PCM output data
- PCM_IN: PCM input data

These four pins are able to share IIS interface follow this mapping:

- PCM_CLK share with IIS_SCLK
- PCM_IN share with IIS_SDI
- PCM_OUT share with IIS_SDO
- PCM_SYNC share with IIS_LRCK

The data can be linear PCM (13-16 bit), μ -Law (8 bit) or A-Law (8bit). The interface can work as either Master or Slave.

- Programming IIS_CTL 0 register bit 15 to 1 configures the PCM mode.

Long Frame Sync is the name given to a clocking format that controls the transfer of PCM data words or samples. In Long Frame Sync, the rising edge of PCM_SYNC indicates the start of the PCM word. When Base Band Chip is configured as PCM Master, generating PCM_SYNC and PCM_CLK, 8 bits in one frame,

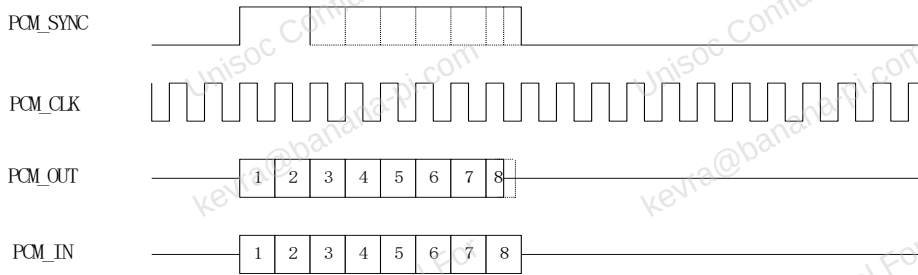
then PCM_SYNC is 7-bit long. When Base Band chip is configured as PCM Slave, PCM_SYNC may be from one consecutive falling edges of PCM_CLK to not big or equal than PCM data length (as figure shows, PCM_SYNC is 2~7.5 bits) long.

- Programming IIS_CTL0 register bit8 to zero configures the Long Frame Sync mode.

PCM_IN is captured on the falling edge of PCM_CLK and PCM_OUT transmits on the rising edge.

PCM_OUT may be configured to be high impedance on the falling edge of PCM_CLK in the LSB position or on the rising edge. The control bit is IIS_CTL2 bit10, if high, OE changes to low at falling edge of the last bit.

Figure 9-54 Long frame sync (Shown with 8-bit sample)

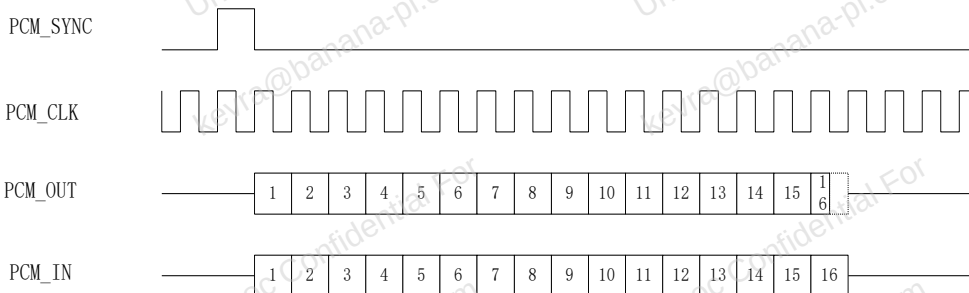


In Short Frame Sync the falling edge of PCM_SYNC indicates the start of the PCM word. PCM_SYNC is always one clock cycle long.

As with Long Frame Sync, Base Band Chip samples PCM_IN on the falling edge of PCM_CLK and transmits PCM_OUT on the rising edge. PCM_OUT may be configured to be high impedance on the falling edge of PCM_CLK in the LSB position or on the rising edge.

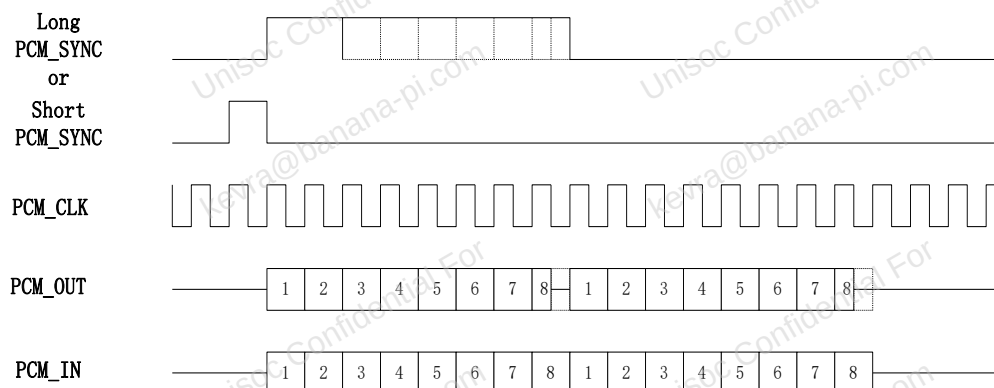
Programming IIS_CTL0 register bit8 to high configures the Short Frame Sync mode.

Figure 9-55 Short frame sync (Shown with 16-bit sample)



More than one channel connection over the PCM interface is supported using multiple slots. Up to three channel connections can be carried over any of the first three slots.

Figure 9-56 Multi-slot operation with two slots and 8-bit compounded samples



Three slots can be configured in one PCM frame; IIS_CTL2 register bit 2~0 define which channel is occupied. Each bit configures one of the three channels. If bit0 is high, then slot0 is carrying data. If channel 0 and 2 is active, this register is 0x5.

PCM data format in TX/RX buffer is different between master and slave mode. If burst size is 16 or 8bit, in slave mode, data for slot 0 and slot 1 are packed into one word, which locates in lower address; data for slot2 locates in higher address; slot0 and slot2 fill higher bits (bit31~24). In master mode, only slot0 takes lower address and locates on lower bits (bit15~8); slot1, slot2 packed in higher address, and slot2 takes lower bits while slot1 in higher bits. Following table shows PCM data format for master and slave mode.

Table 9-15 PCM data format for master mode

IIS_CTL bit1~0	Bit/channel	Data arrangement
00	8bit	Data for channel 0 on bit 15~8, in lower address. Data for channel 1 on bit 31~24, Data for channel 2 on bit 15~8, on higher address.
01	16bit	Data for channel 0 on bit 15~0, in lower address. Data for channel 1 on bit 31~16, Data for channel 2 on bit 15~0, on higher address
1x	32bit	Data for channel 0 store in low address, data for channel 1, 2 store in high address.

Table 9-16 PCM data format for slave mode

IIS_CTL bit1~0	Bit/channel	Data arrangement
00	8bit	Data for channel 0 on bit 31~24, Data for channel 1 on bit 15~8, in lower address. Data for channel 2 on bit 31~24, on higher address.

IIS_CTL bit1~0	Bit/channel	Data arrangement
01	16bit	Data for channel 0 on bit 31~16, Data for channel 1 on bit 15~0, in lower address. Data for channel 2 on bit 31~16, on higher address
1x	32bit	Data for channel 0 store in low address, data for channel 1, 2 store in high address.

Table 9-17 PCM multiple slot and PCM cycle configuration suggestion

pcm_slot[0]	pcm_slot[1]	pcm_slot[2]	pcm_cycle
0	0	1	<=2
0	1	0	>=1
0	1	1	>=1
1	0	0	>=2
1	0	1	>=2
1	1	0	>=2
1	1	1	>=2

Table 9-18 Exceptions for PCM multiple slot configuration

Bit width	pcm_slot[0]	pcm_slot[1]	pcm_slot[2]	pcm_cycle	sample rate	clk_iis
32	1	-	1	2	8 or 44.1 or 48 kHz	26/80MHz
16,8				4		

NOTE

In PCM fraction divide mode, clk_iis frequency should be 20 times bigger than bit clock

For EDA simulation, pcm_slot[2:0]=101, bit_width=8 or 16, pcm_cycle>=2 is ok, but software test meet some problem when sample rate > 1kHz. So suggest software engineer setting pcm_cycle=4 in application. It's ok for both software and EDA simulation.

9.10.3.3 IIS clock

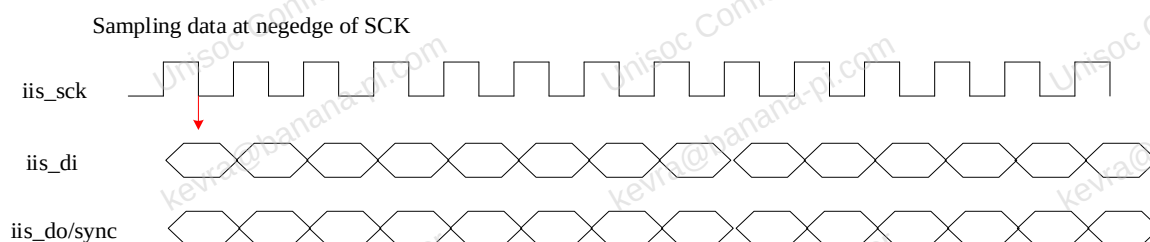
- Module clock: iis_clk, PCLK.
- IO pad: iis_sck, iis_sync, iis_di, iis_do.
- IIS IP can be configured as master mode or slave mode. The configure signal is iis_salved
- If iis_salved= 1, IIS works as slave. iis_m_top will not work.
- If iis_salved = 0, IIS works as master. iis_s_top will not work.
- “divider” block function: iis_clk_p = iis_clk/M. where M is a clock divider factor.

- “reverse” block function: $iis_clk_o = \sim iis_clk_p$.
- “sck counter” block function: count iis_sck period. Count unit is iis_clk .
- “double sync” block is used to resolve crossing clock domain signal.

9.10.3.4 Pad IO timing

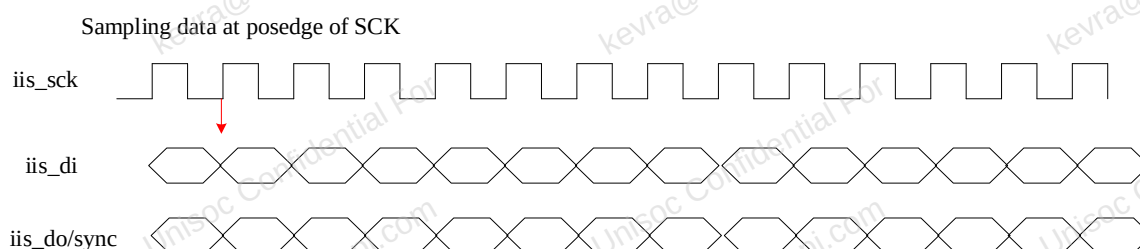
In default mode, IIS output signals, like iis_do , iis_sync (for master) signal, are sent at the posedge of iis_sck . And input signals like iis_di , iis_sync (for slave), are sampled at the negedge of iis_sck .

Figure 9-57 Sampling data at negedge of IIS_SCK



You can also set output signal sent edge as negedge and sampling edge as posedge, like the following.

Figure 9-58 Sampling data at posedge of IIS_SCK



The implementation of sampling edge is as following.

Figure 9-59 implementation of sampling data at negedge of IIS_SCK

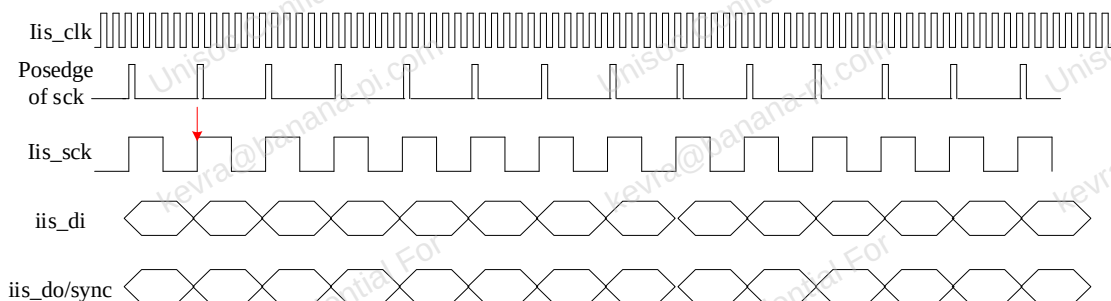
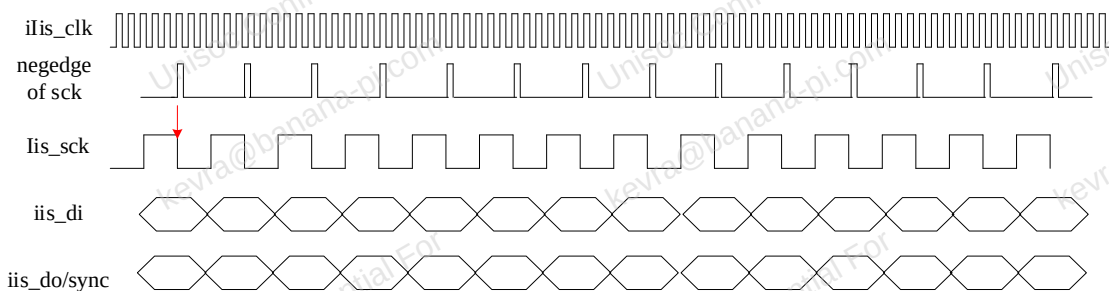


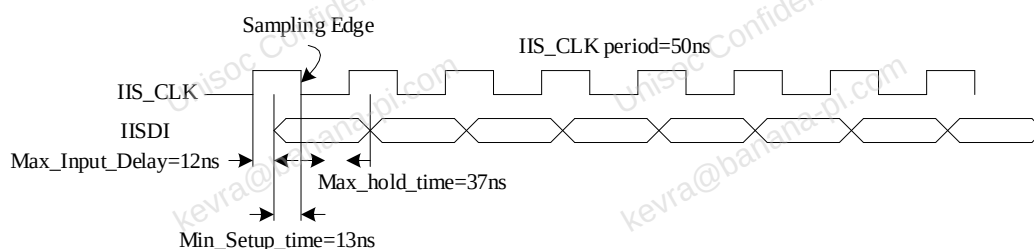
Figure 9-60 implementation of sampling data at posedge of IIS_SCK



9.10.3.5 Setup time and hold time

If assume max input delay is 10ns and min input delay is 0ns for iis_di to iis_sck. max output delay is 10ns and min output delay is 5ns for iis_do/iis_lrck to iis_sck. The clock period time is 50ns for iis_sck.

Figure 9-61 implementation of sampling data at posedge of IIS_SCK



Then for iis_di, the setup time range is $13\text{ns} < \text{setup_time} < 25\text{ns}$, and the hold time range is $25\text{ns} < \text{hold_time} < 37\text{ns}$. For iis_do/iis_lrck, the setup time range is $13\text{ns} < \text{setup_time} < 25\text{ns}$, and the hold time range is $25\text{ns} < \text{hold_time} < 37\text{ns}$.

9.10.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.10.5 Application Notes

9.10.5.1 IIS Program examples

- Justified format, master, 16 bits per-channel, LRCK low for left, Program control register:
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_08d1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0000
- IIS format, master, 8 bits per-channel, LRCK high for left, Program control register:
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_0dc1
 - iis_ctl1 = 32'h0000_0f11

- iis_ctl2 = 32'h0000_0000
- DSP (sync) format, master, 16 bits per-channel, MSB is available on the 1st rising edge, Program control register
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_0ad1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0000
- DSP format, master, 16 bits per-channel, MSB is available on the 1st rising edge, Program control register:
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_0ed1
 - iis_ctl1 = 32'h0000_3f11
 - iis_ctl2 = 32'h0000_7800
- Justified mode, slave, 8 bits per-channel, enable clock counter, Program control register:
 - iis_ctl0 = 32'h0000_08c9
 - iis_ctl1 = 32'h0000_3f22
 - iis_ctl2 = 32'h0000_0000

9.10.5.2 PCM program examples

Setting IIS_CTL0 register bit[15] to “1” enables IIS module working in PCM mode.

- Long Frame format, master, 8 bits per-channel, RX at falling edge, TX at rising edge, Program control register
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_80c1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0001
- Short Frame format, master, 8 bits per-channel
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_81c1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0001
- Multi-cycle format, master, 8 bits per-channel, pcm_oe=1, pcm_cyc=1, slot0 is occupied
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_80c1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0409
- Multi-cycle format, master, 8 bits per-channel, pcm_cyc=2, slot0 and slot1 are occupied
 - iis_clkd = 32'0000_000a
 - iis_ctl0 = 32'h0000_80c1
 - iis_ctl1 = 32'h0000_0f11
 - iis_ctl2 = 32'h0000_0013
- Long frame format, slave, 16 bits per-channel, enable clock count
 - iis_ctl0 = 32'h0000_80c9

- iis_ctl1 = 32'h0000_3f11
- iis_ctl2 = 32'h0000_0001
- Short frame format , slave, 16 bits per-channel,enable clock count
 - iis_ctl0 = 32'h0000_81cd
 - iis_ctl1 = 32'h0000_3a1
 - iis_ctl2 = 32'h0000_0001

9.10.5.3 Clock dividing modes

In master mode, the IIS serial clock iis_sck is sent from the IIS module, and this clock is divided from clk_iis. IIS module provides three modes for clock dividing.

- Integer clock dividing mode
 - Both IIS and PCM master support this mode.
 - This mode can be used only if iis_sck is integrally divided from clk_iis.
 - Mode enable: IIS_CLK_DVDR_MD0 = 0 and IIS_CLK_DVDR_MD1 = 0
 - Only IIS_CLKD is used for dividing; IIS_CLKM and IIS_CLKN are DON'T CARE.
 - $IIS_CLKD = (clk_iis / (2 \times iis_sck)) - 1$; $iis_sck = \text{sample rate} \times \text{channel number} \times \text{channel length}$

For example, if clk_iis runs at 153.6MHz, and IIS module is configured as 48k sample rate, 2 channel and 16-bit channel length:

 - iis_sck should be $48k \times 2 \times 16 = 1.536MHz$
 - IIS_CLKD should be $(153.6M / (2 \times 1.536M)) - 1 = 49$
- Fraction clock dividing mode0
 - Both IIS and PCM master support this mode.
 - In this mode, the actual frequency of iis_sck is variable, and only average is target frequency. Because of this variability, the iis_lrck frequency is also variable, and only average is target sample rate.
 - Mode enable: IIS_CLK_DVDR_MD0 = 1 and IIS_CLK_DVDR_MD1 = 0
 - IIS_CLKM and CLKN are used for dividing; while IIS_CLKD is not cared.
 - $IIS_CLKM / IIS_CLKN = 2 \times (\text{target } iis_sck) / clk_iis$; $(\text{target } iis_sck) = \text{sample rate} \times \text{channel number} \times \text{channel length}$
 - $IIS_CLKM = 2 \times \text{sample rate (Hz)} \times \text{channel number} \times \text{channel length} / 100$
 - $IIS_CLKN = clk_iis \text{ (Hz)} / 100$

For example, if clk_iis runs at 153.6MHz, and IIS module is configured as 44.1kHz sample rate, 2 channel and 16-bit channel length:

 - $IIS_CLKM = 2 \times 44100 \times 2 \times 16 / 100 = 28224$
 - $IIS_CLKN = 153600000 / 100 = 1536000$
- Fraction clock dividing mode1
 - Only IIS master supports this mode.
 - In this mode, the frequency of iis_sck is fixed. But the actual sample rate is variable, and only average is target sample rate.
 - Compared with MD0, the advantage of MD1 is that iis_sck is fixed, but the disadvantage is that the jitter of sample rate is higher.
 - Mode enable: IIS_CLK_DVDR_MD0 = 0 and IIS_CLK_DVDR_MD1 = 1
 - IIS_CLKM, CLKN and CLKD are used for dividing
 - $IIS_CLKD = (clk_iis / (2 \times iis_sck)) - 1$

- $IIS_CLKM / IIS_CLKN = (\text{target sample rate}) \times \text{channel number} \times \text{channel length} / iis_sck$; IIS_CLKM must be not more than IIS_CLKN .
- $IIS_CLKM = 2 \times (IIS_CLKD + 1) \times (\text{target sample rate}) \times \text{channel number} \times \text{channel length} / 100$
- $IIS_CLKN = clk_iis / 100$
- The iis_sck (or IIS_CLKD) should be decided by software. iis_sck must be not lower than $((\text{target sample rate}) \times \text{channel number} \times \text{channel length})$. And iis_sck should be not too high for timing issue. Usually iis_sck is a little bit higher than $((\text{target sample rate}) \times \text{channel number} \times \text{channel length})$.

For example, if clk_iis runs at 153.6MHz, and IIS module is configured as 44.1kHz sample rate, 2 channel and 16-bit channel length. Firstly, choose iis_sck as $153.6M / (2 \times (44 + 1)) = 1.706667MHz$ because $((\text{target sample rate}) \times \text{channel number} \times \text{channel length}) = 1.4112MHz$ and 1.706667MHz is a little bit higher than 1.4112MHz.

- $IIS_CLKD = 44$
- $IIS_CLKM = 2 \times (44 + 1) \times 44100 \times 2 \times 16 / 100 = 1270080$
- $IIS_CLKN = 153600000 / 100 = 1536000$

Some guidelines for clock dividing modes:

- Integer clock dividing mode is preferred if possible, because this mode has fixed serial clock frequency and fixed sample rate.
- If iis_sck cannot be integrally divided from clk_iis , choose fraction clock dividing modes. If system needs fixed serial clock frequency, mode1 is chosen. Otherwise mode0 is preferred for small sample rate jitter.
- IIS master supports both mode0 and mode1, and PCM master supports only mode0.
- The frequency options of clk_iis depend on different chips. But 153.6MHz is always preferred because this frequency can be divided integrally to support 48k, 24k, 12k, 32k, 16k and 8k sample rate.
- Fraction clock divide mode1 configure.

Table 9-19 IIS configure for fraction clock divide

MD0, MSB-justified	Master	Slave
IIS_CTL[5]	0	x(don't care)
IIS_CTL[4]	1	x(don't care)
IIS_CTL[1]	x(don't care)	x(don't care)
IIS_CTL[0]	x(don't care)	x(don't care)

MD0, compatible	Master	Slave
IIS_CTL[5]	0	x(don't care)
IIS_CTL[4]	1	x(don't care)
IIS_CTL[1]	x(don't care)	x(don't care)
IIS_CTL[0]	x(don't care)	x(don't care)

MD1, MSB-justified	Master	Slave
IIS_CTL[5]	1	x(don't care)
IIS_CTL[4]	0	x(don't care)
IIS_CTL[1]	x(don't care)	0
IIS_CTL[0]	x(don't care)	1

MD1, compatible	Master	Slave
IIS_CTL[5]	1	x(don't care)
IIS_CTL[4]	0	x(don't care)
IIS_CTL[1]	x(don't care)	1
IIS_CTL[0]	x(don't care)	0

Table 9-20 iis_clk configure suggestion

IIS mode	rate	width	clk_iis
compatible	192k	32	153.6M(Suggest)
compatible	192k	32	128M(Suggest)
compatible	192k	32	80M(Not Suggest)
compatible	192k	32	26M(Not Suggest)
compatible	192k	16	153.6M(Suggest)
compatible	192k	16	128M(Suggest)
compatible	192k	16	80M(Suggest)
compatible	192k	16	26M(Not Suggest)
compatible	192k	8	153.6M
compatible	192k	8	128M
compatible	192k	8	80M
compatible	192k	8	26M
compatible	44.1k	32	153.6M
compatible	44.1k	32	128M
compatible	44.1k	32	80M
compatible	44.1k	32	26M
compatible	44.1k	16	153.6M

IIS mode	rate	width	clk_iis
compatible	44.1k	16	128M
compatible	44.1k	16	80M
compatible	44.1k	16	26M
compatible	44.1k	8	153.6M
compatible	44.1k	8	128M
compatible	44.1k	8	80M
compatible	44.1k	8	26M
Justified	192k	32	153.6M(Suggest)
Justified	192k	32	128M(Suggest)
Justified	192k	32	80M(Not Suggest)
Justified	192k	32	26M(Not Suggest)
Justified	192k	16	153.6M(Suggest)
Justified	192k	16	128M(Suggest)
Justified	192k	16	80M(Suggest)
Justified	192k	16	26M(Not Suggest)
Justified	192k	8	153.6M
Justified	192k	8	128M
Justified	192k	8	80M
Justified	192k	8	26M
Justified	44.1k	32	153.6M
Justified	44.1k	32	128M
Justified	44.1k	32	80M
Justified	44.1k	32	26M
Justified	44.1k	16	153.6M
Justified	44.1k	16	128M
Justified	44.1k	16	80M
Justified	44.1k	16	26M
Justified	44.1k	8	153.6M
Justified	44.1k	8	128M
Justified	44.1k	8	80M
Justified	44.1k	8	26M

NOTE

In IIS fraction clock divide model, clk_iis frequency should be higher when bit clock is fast.

9.10.5.4 Working mode examples

- DMA mode, PCM, Slave, 16bit

```
iis_clkd = 32'0000_000a
iis_ctl0 = 32'h0000_c3d9
iis_ctl1 = 32'h0000_0f11
iis_ctl2 = 32'h0000_0001
iis_ctl3 = 32'h0000_0f0f (threshold is half volume of buffer)
iis_int_en=32'h0000_0036
```

- Interrupt mode, PCM, slave, 16bit

```
iis_clkd = 32'0000_000a
iis_ctl0 = 32'h0000_83d9
iis_ctl1 = 32'h0000_0f11
iis_ctl2 = 32'h0000_0001
iis_ctl3 = 32'h0000_0f0f (threshold is half volume of buffer)
iis_int_en=32'h0000_00f6
```

NOTE

IIS is same configuration as pcm mode.

9.10.5.5 Debug tips

To ensure that it is working well, the first step is to test iis_sck, iis_lrck, iis_di, iis_do using scope or by observing state of IIS_STS2.

In the IIS mode, iis_lrck changes periodically, and high/low duty ratio is 50%. In the pcm mode, iis_lrck is a periodical pulse.

If iis_sck, iis_lrck, iis_di, iis_do is normal, then you can operate IIS_TXD. Transmitting a word by writing data to IIS_TXD in one side (e.g. transmitter). Observe the result by reading data from IIS_TXD in another side (e.g. receiver).

IIS working on fraction mode1 and channel width = 32bit, IIS_DUMMY_DEBUG is set to be 1. And in other cases, IIS_DUMMY_DEBUG should set to be 0. IIS_DUMMY_DEBUG bit is added.

9.11 UART**9.11.1 Overview**

UART is an asynchronous communication interface. The UART module includes a baudrate generator with software-programmable divider ratios for all common baudrates.

Two types deep FIFO are included. Receive FIFO which is 128-Byte-deep and send FIFO which can be configured to 128-Byte-deep or 256-Byte-deep; The module also includes a flexible interrupt with multiple maskable interrupt sources. Two hardware flow control lines are included (one for input, the other for output). This module also has a built-in IrDA controller and can be configured to connect to an IrDA transceiver. When being used as regular UART port, the IrDA controller is bypassed.

9.11.2 Features

- Full-duplex operation
- Hardware flow control support
- Software flow control support with configurable command and escape register
- Configure TX FIFOs minimizes 128-Byte-deep or 256-Byte-deep processor overhead at high data rates
- Wide selection of UART word lengths, including 5,6,7 and 8 bits
- Stop bit number can be 1,1.5 or 2 bits
- RX stop bit number and TX stop bit number can be configured separately
- Automatic baudrate detection flow support
- Support odd/even parity
- Auto detect for parity and framing error
- UART loop-back test mode
- Break character detection and generation
- support DMA operation
- Single interrupt line for multiple maskable interrupt source events
- Programmable interrupt trigger levels for FIFOs
- Baudrate generation based upon programmable divisors, operating from a flexible functional clock
- Includes a built-in IrDA controller
- Support 32 bits or 8 bits write once to txfifo.

9.11.3 Function Description

The UART and IRDA baudrate generator is based on the `clk_uart`, the divisor coefficient is controlled by register `UART_CKD0`. IRDA part clock period is the 1/16 of the UART part since the required timing of IRDA is 3/16 or 4/16 of bit width. The serial data rate can be changed by modify the `clk_uart` or the UART divisor coefficient.

The UART module has separated RX and TX channel to realize the duplex working mode. And the UART module also supports to enable the IRDA transceiver mode.

9.11.3.1 UART Mode Description

UART module has an independent baudrate generator, which is used as a reference clock for data transmit and recovery. It also has a 128-Byte-deep RX FIFO and a 128-Byte-deep TX FIFO to decrease processor interrupt load during the data receiving and transmitting.

9.11.3.2 FIFO Management

FIFOs are accessed by reading/writing `UART_RXD/UART_TXD` register. The number of data in TX/RX FIFO can be observed by reading `UART_STS1 [6:0]/UART_STS1 [15:8]`. `Rx_fifo_full` and `Tx_fifo_empty` status can be obtained by reading `UART_STS0 [1:0]`. `UART_CTL2` register controls the FIFO trigger level, which enables the DMA request and interrupts generation.

- TX FIFO size is 128 Bytes or 256 Bytes. It uses a FIFO pointer to monitor the number of data in the TX FIFO that has not yet been transmitted. The MCU can read the `UART_STS1 [15:8]` to access the pointer. The UART also provides a TXF empty interrupt threshold that can be configured by writing `UART_CTL2 [15:8]`.
- RX FIFO size is 128 Bytes. It uses a pointer to monitor the number of data in the RX FIFO that have not yet been read by the MCU. The MCU can access the pointer by reading the `UART_STS1 [6:0]`. The UART also provides a RX FIFO full interrupt threshold that can be configured by writing to `UART_CTL2 [6:0]`. In FIFO interrupt mode with flow control, the programmer must also ensure that the

Receive_Hardware_Flow_Control_Threshold (UART_CTL1 [6:0]) value is greater than or equal to the Receive_Int_Threshold (UART_CTL2 [6:0]). Otherwise, FIFO operation stalls.

In FIFO interrupt mode, the processor is informed of the status of the receiver and transmitter by an interrupt signal. The rx_fifo_full interrupt is set when the number of RX FIFO data Byte is larger than the RX interrupt full threshold (threshold value). It is auto cleared when the condition disappears. The tx_fifo_empty interrupt is set when the number of TX FIFO data Byte is less than the TX interrupt empty threshold (threshold value). It is auto cleared when the condition disappears. The two interrupt sources can be masked by writing “0” to UART_IEN register bit 0 and bit 1. The interrupt signal instructs the local hosts to write data to TX FIFO or read data from RX FIFO.

Note that in the cases of the UART flow control being enabled along with the interrupt capabilities, the user must ensure that the UART flow control FIFO threshold is greater than or equal to the receive FIFO threshold.

The TX and RX FIFO may be accessed by DMA. Writing to control register UART_CTL1 bit15 will enable DMA mode. In receive mode, a DMA request is generated as soon as the receive FIFO reached its threshold level defined in UART_CTL2 [6:0]. This request is deserted when the number of Bytes is less than the threshold level. In transmit mode, a DMA request is automatically asserted when the transmit FIFO is almost empty.

In this mode, when RX FIFO is almost full (the number of RX FIFO data Byte is larger than the RX interrupt threshold value), it will send the receive request to DMA controller, and when TX FIFO is almost empty, it will send the transmit request to DMA controller.

9.11.3.3 Flow Control

The UART module support hardware and software flow control. The hardware flow control should use rtsn/ctsn signal to indicate the flow status. The software flow control use command which integrated in data stream in the txd/rxd signal to indicate the flow control status. The hardware flow control has two mode, 1) auto flow control, 2) software configure hardware flow control. When the UART_SFCTL register bit1 is set to 1, the software flow control enabled, or if the UART_SFCTL[1] is 0 and if the UART_CTL1[8:7] is set to 2'b11 the auto hardware flow is enabled, or if the UART_CTL[8:7] is 2'b00, the rtsn is set by software, and the receive flow control is disabled. But UART can't literally switch flow control mode when UART start to work.

9.11.3.3.1 Hardware Flow Control

The UART module supports both auto hardware flow control and software configured hardware flow control. When UART_CTL1 register bit 7 is set to “0”, software will control the signal uart_rtsn by writing UART_CTL0 register bit 6; if “1”, the signal uart_rtsn will be controlled by hardware itself, called auto-RTS.

Auto hardware flow control is composed of auto-CTS and auto-RTS. Auto-CTS and auto-RTS can be enabled/disabled independently by programming UART_CTL1 [8:7].

Auto-RTS data flow control originates in the receiver block. The receiver FIFO trigger levels used in auto-RTS are stored in the UART_CTL1 [6:0]. RTS is active if the RX FIFO level is below the HALT trigger level. When the receiver FIFO HALT trigger level is reached, uart_rtsn is deserted. The sending device (external UART device) may send an additional Byte after the trigger level is reached because it may not recognize the desertion of RTSn until it has begun sending the additional Byte. The assertion requests the sending device to resume transmission. In this case, uart_rtsn is an active-low signal.

With auto-CTS, uart_ctsn must be active before the module can transmit data. The transmitter circuitry checks uart_ctsn before sending the next data Byte. When uart_ctsn is active, the transmitter sends the next Byte. The auto-CTS function reduces interrupts to the host system. When auto-CTS flow control is enabled, the CTS state changes need not trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result. In this case, uart_ctsn is an active-low signal.

9.11.3.3.2 Software Flow Control without CTSn and RTSn

The UART module support software flow control without use CTSn and RTSn signals. When the UART_SFCTL register **bit[1]** is set to 1, the software flow control is enabled. The user also can config the UART_SFCTL register **bit[0]** to select the mode of the software flow control function.

The receiver endpoint use the receive FIFO index to judge whether to suspend the transmission. When the receive FIFO index is larger than the upper threshold CTRL1[6:0] (the same as hardware flow control threshold), the receive endpoint will send XOFF command by `uart_tx` to the transmit endpoint, then the transmit will be suspend. When the receive FIFO index is smaller than the lowerthreshold DSPWAIT[16:10], the receive endpoint will send the XON command to transmitter to restart the transmission. When configure the register must guarantee the upper threshold CTRL1[6:0] bigger or equivalent the lower threshold DSPWAIT[16:10]. Vice verse, when transmitting data, the `uart_rx` will receive the XOFF and XON command to suspend and restart the transmission. So the UART module will realize the control flow without use CTSn and RTSn signals, it can reduce the usage of IO resource.

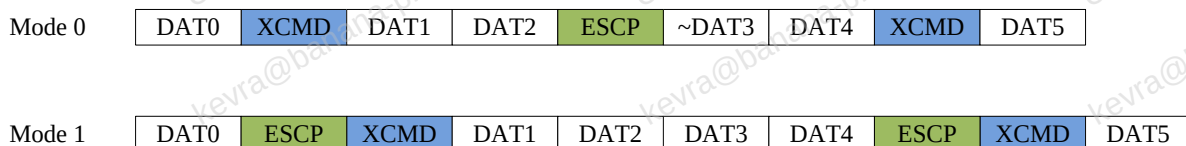
Since the software flow control will send command to represent the flow control status, there will be several data have the same value as the XCMD(XON/XOFF), in order to prevent transmission error, the data or the command should be escaped before transmitting or after the receiving. So it will use another command ESC to represent the data status.

The UART module supports two modes for escaping. The first mode is send XCMD as the normal data, and use ESC command and the inverted data to represent the transfer data that has the same value of the XCMD and ESC. The second kind is send the data that has the same value of the XCMD as normal uart transmitting, and send XCMD with a ESC prefix. The format example is shown as follow.

Suppose using 0x11 to represent XON, 0x13 to represent XOFF, 0x77 to represent ESC:

- When use mode 0 to transmit data 0x11 0x13 0x77, the data should be escape to 0x77 0xEE 0x77 0xEC 0x77 0x88. When transmit XON, it should be transmitting 0x11.
- When use mode 1 to transmit data 0x11 0x13, the data can be transmitted without escape, when transmit XON, the transmit data should be 0x77 0x11, when transmit data 0x77, the data should be escaped to 0x77 0x77.

Figure 9-62 Software Flow Control Data Stream (Mode0/Mode1)



9.11.3.4 Interrupt sources

UART provides a TXF empty interrupt threshold that can be configured by writing UART_CTL2 [14:8]. A `tx_fifo_empty` interrupt will be generated when the number of TX FIFO data Byte is less than the TX interrupt threshold value. It is auto cleared when the condition disappears.

UART also provides a RX FIFO full interrupt threshold that can be configured by writing to UART_CTL2 [6:0]. An `rx_fifo_full` interrupt will be generated when the number of RX FIFO data Byte is larger than the RX interrupt threshold value. The interrupt will be auto cleared when the condition disappears.

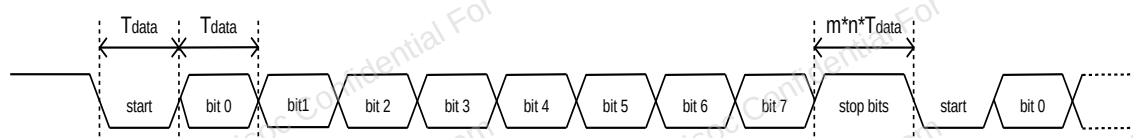
An `rx_tout` interrupt will be generated if the UART has not received data from the input line in a period of time. This period depends on the baudrate and the value of CTL1 [13:9]. If the value of CTL1 [13:9] is '0', the interrupt will never be generated. The interrupt can be cleared by writing "1" to UART_ICLR bit13.

When having received data, UART generate parity error/framing error if error condition occurs. Also, when having received FIFO overrun, an error interrupt will be generated. Writing to corresponding interrupt clear bit will clear these interrupts.

All interrupt sources in UART module can be disabled and masked by writing to the corresponding registers.

When TX FIFO is empty and TX is idle, setting send break bit forces the TX data output to low.

Figure 9-63 UART timing parameter



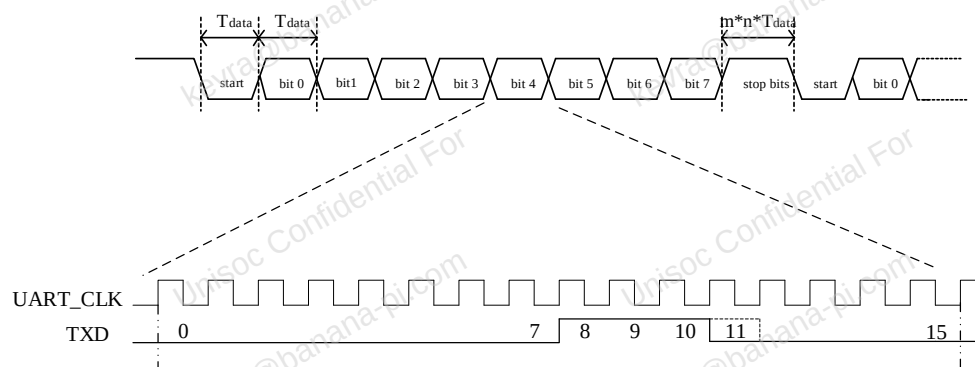
9.11.3.5 IrDA Mode Description

The IrDA can be used to communicate with other IrDA compatible device. IrDA mode is selected by writing “1” to UART CTL0 [15]. The default value of this bit is “0”.

IrDA mode is achieved by adding IrDA transmit block and IrDA receive block into UART module. The IrDA transmit block changes the UART transmit serial bits to the IrDA timing, and then sends to TXD port. On the other side, the receive module detects the IrDA timing from the RXD port and changes back to UART bit stream to let the UART receive block to finish the data receiving.

Figure 9-64 below shows the transmit IrDA timing. The polarity could be changed by UART CTL0 bit 9, and the pulse of a bit may be 3 or 4 high cycle controlled by the UART CTL0 bit13. In the no polarity invert working mode, when sending “1”, the output remains “0”. And when sending “0”, output follows the IrDA timing as demonstrated in the figure. On the receive data path, the polarity could be change by UART_CTL0 bit 10. The transmission and receive operation could be controlled by the UART CTL0 bit11 and bit12. UART_CKD0[15:4] must be set bigger than UART_CKD0[3:0] when using irda mode.

Figure 9-64 IrDA transmit timing



9.11.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.11.5 Application Notes

Automatic baudrate detection flow

1. The initiator writes to the setup register (global register).
2. The initiator writes UART_IEN register bit 16 to “1” to enable ABD done interrupt.
3. The initiator writes UART_CTRL1 bit 19 to “1” to enable ABD mode.
4. ABD done interrupt will be generated if UART receive the ASCII code of “at” or “AT”. After receiving this interrupt, MCU should clear the interrupt and configure the UART_CTL0 to generate the suitable data format by the information of UART_ABD0, UART_ABD1 and UART_ABD2 registers.
5. Writes UART_CTRL1 bit 19 to “0” to disable ABD mode and enter the UART mode.

Working in UART mode with hardware flow control (full duplex)

1. The initiator writes to the setup register (global register).
2. The initiator writes the UART_CTL0 register bit 15 to “0” to enable UART mode.
3. The initiator configures UART_CKD0 registers to generate the suitable baudrate. For example, if clk_uart is 26MHz, UART_CKD0 is set to 0xe1, then the UART baudrate is 115kHz.
4. The initiator configures UART_CTL0 to generate the suitable data format, such as parity enable and parity style, Byte length, and stop bit number.
5. Set UART_CTL1 register bit 7 to “1” to enable receive hardware flow control.
6. The initiator configures UART_CTL1 register bit [6:0] to suitable value.
7. Set UART_CTL1 register bit 8 to “1” to enable transmit hardware flow control.
8. The initiator configures UART_CTL2 to generate receive threshold and transmit threshold.
9. The initiator writes UART_CTL1 register to configure receive timeout value.
10. Enable related interrupts by writing UART_IEN register.
11. Writing data to UART_TXD.

Once the TX FIFO is not empty:

1. Data in TX FIFO will appear on TX line in sequence.
2. If data number in the TX FIFO is less than the data empty threshold value, a txf_empty interrupt will be generated. After receiving this interrupt, MCU should write more data into the TX FIFO. After data number in the TX FIFO is more than the data empty threshold value, MCU should clear the interrupt.
3. If data number in the RX FIFO is more than the data full threshold value. An rxf_full interrupt will be generated. After receiving this interrupt, MCU should read data from RX FIFO. After data number in the RX FIFO is less than the data full threshold value, MCU should clear the interrupt.
4. If receiving RX_TOUT interrupt, read all data in the RX FIFO.

When use the software flow control without ctsn and rtsn signal (full duplex)

1. The initiator writes to the setup register (global register).
2. The initiator writes the UART_CTL0 register bit 15 to “0” to enable UART mode.
3. The initiator configures UART_CKD0 registers to generate the suitable baudrate. For example, if clk_uart is 26MHz, UART_CKD0 is set to 0xe1, then the UART baudrate is 115kHz.
4. The initiator configures UART_CTL0 to generate the suitable data format, such as parity enable and parity style, Byte length, and stop bit number.
5. The initiator configures UART_CTL1 to generate the Recieve Hardware Flow Control Threshold (software flow control also used this value to determine the rx_fifo is full).

6. The XON/XOFF/ESCAPE register should be configured before the enable of software flow control and mode selection.
7. The initiator configures UART_CTL2 to generate the TX FIFO data empty threshold and RX FIFO data full threshold. (must guarantee the value is lower than or equal to Recieve_Hardware_Flow Control_Threshold)
8. The initiator writes UART_CTL1 register to configure receive timeout value.
9. The initiator configures UART_IEN to make TX FIFO empty and RX FIFO full interrupt enable.
10. The software flow control enable bit should be configured before starting a software flow control transition.
11. Writing data to UART_TXD.
12. When UART_IRQ set, reading data from UART_STS2 to configure the related interrupt.

9.12 Spinlock

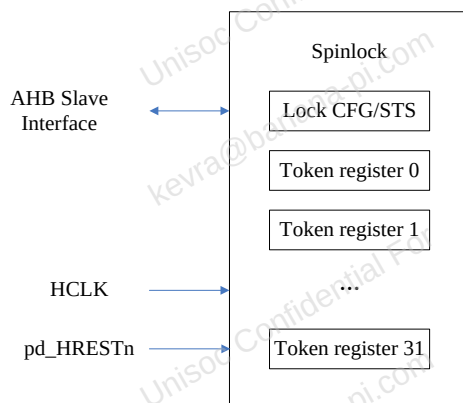
9.12.1 Overview

Hardware spinlocks are provided to assist multiple processors to synchronize access to global resources.

9.12.2 Features

- Support 32 spinlocks
- Use special token(0x0000_0000) indicate unlock
- Perform lock operation by reading Spinlock Token Registers
- Perform unlock operation by writing Unlock Token(0x55AA10C5) to Spinlock Token Registers
- Spinlock Total Status Register is just for debug tool, takes no effect on spinlock

Figure 9-65 Spinlock



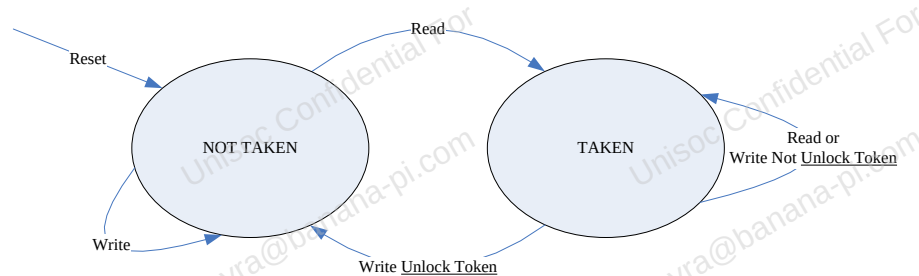
9.12.3 Function Description

9.12.3.1 Take and Release Spinlock

- The spinlock has two states: **Taken** and **Not Taken**.
- After reset, the state of lock is **Not Taken**, equal to 0x0000_0000.

- When the lock is in **Not Taken** state, all write takes no effect.
- When the lock is in **Not Taken** state, a read returns 0x0000_0000, and change to **Taken** state.
- When the lock is in **Taken** state, all write except the Unlock Token(0x55AA10C5) takes no effect. And a read always returns 0x0000_0001.
- A write of the Unlock Token sets the lock back to **Not Taken** state.

Figure 9-66 take and release spinlock



NOTE

There is not support to ensure that a lock is locked and unlocked by the same master by default. This must be ensured in software.

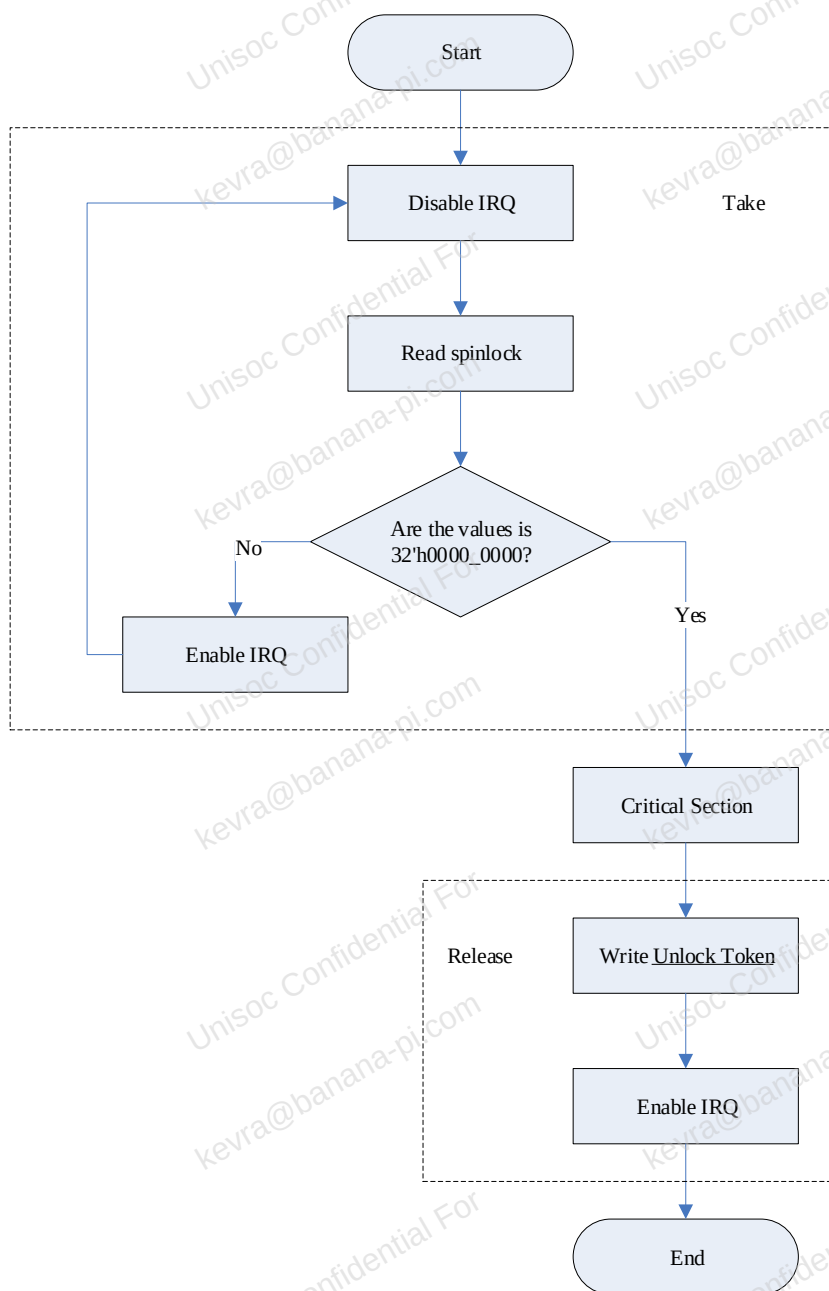
9.12.3.2 Programming Guide

A spinlock should only be held with interrupts disabled. So, before attempting to obtain the spinlock, software should disable interrupts.

Then it should read spinlock. Reading back 0x0000_0000 indicates success. Now it should process the critical section then write Unlock Token(0x55AA10C5) unlock and re-enable interrupts. If the acquisition attempt fails, the acquisition should be reattempted.

To prevent unknown interrupt disabled time, interrupts should be re-enabled and then disabled before reattempting to acquire the lock.

Figure 9-67 spinlock take and release flow



9.12.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.13 Watchdog timer

9.13.1 Overview

Watchdog timer is a 32-bit decreasing timer with RTC-clock triggered, and this timer can be used to issue system reset.

9.13.2 Features

- RTC-clock triggered
- 32-bit decreasing timer
- Support 32-bit timer value loading
- Support reset mode for watchdog function
- Support interrupt mode for debug function and timer function
- Support combined mode: first generate interrupt, and then generate reset.
- Support single read: no need to double read when reading counter value
- Support multiple load: no need to check busy bit before load wdg_cnt
- Support read counter value immediately after loading wdg_cnt: once wdg_cnt is loaded, the counter value is updated immediately

9.13.3 Function Description

After hardware reset, watchdog timer is 32'hFFFF_FFFF.

And after enabled, watchdog timer decreases from the reset value or from software-loading value.

The counting step is one RTC clock cycle.

For reset mode, whenever watchdog timer gets to 0, system reset is issued. Then system reboots, and watchdog timer comes back to 32'hFFFF_FFFF, and all controlled registers and status also come back to reset value, except interrupt raw status. Interrupt raw status can be used to judge if or not system rebooting comes from watchdog reset.

For interrupt mode, an interrupt is issued when watchdog timer gets to wdg_irq_value, that is {wdg_irq_value_high, wdg_irq_value_low}. Then the timer continues to count down until it reach 0. After that, watchdog timer comes back to the loaded value.

Combined mode is the combination of reset mode and interrupt mode. Interrupt will be generated first if gets to WDG_IRQ_VALUE, that is {wdg_irq_value_high, wdg_irq_value_low}. Then software should stop wdg so that wdg reset is not generated. If software cannot response to this interrupt, then wdg reset will be generated finally when wdg_cnt decrease to 0 which reboots the entire system.

After enabled, watchdog timer can be loaded at any time, and software should guarantee the timer is loaded before it decreases to 0.

Reset mode is used as normal watchdog function. And interrupt mode is used as debug mode or as a generic periodic timer.

9.13.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.13.5 Application Notes

When software uses watchdog, it should set `wdg_eb` bit and `wdg_rtc_eb` bit in global control register, and also should guarantee `arch_rtc_eb` is set.

Before configuring watchdog, software should write `16'hE551` to `wdg_lock` to unlock watchdog. Then Software configures all control registers: `wdg_ld_value_low`, `wdg_ld_value_high`, `wdg_irq_value_low`, `wdg_irq_value_high`, `wdg_open` and `wdg_new`.

IMPORTANT: `wdg_ld_value_high` should always be configured before writing `wdg_ld_value_low`. Then `wdg_open` is set to 1 to open counter.

Reset mode is used as normal watchdog function. After enabled, watchdog timer can be loaded at any time, and software should guarantee the timer is reloaded before it decreases to 0.

Whenever watchdog timer gets to 0, system reset is issued. Then system reboots, and watchdog timer comes back to `32'hFFFFFF`, and all controlled registers and status also come back to reset value, except interrupt raw status. Interrupt raw status can be used to judge if or not system rebooting comes from watchdog reset.

For interrupt mode, an interrupt is issued when watchdog timer gets to `WDG_IRQ_VALUE`, that is { `wdg_irq_value_high`, `wdg_irq_value_low` }. Then the timer continues to count down until it reach 0. After that, watchdog timer comes back to the loaded value.

After receiving interrupt, software can check interrupt status by reading `wdg_irq_mask`, and can clear interrupt by writing 1 to `wdg_irq_clr`.

Combined mode is the combination of reset mode and interrupt mode. Interrupt will be generated first if gets to `WDG_IRQ_VALUE`, that is { `wdg_irq_value_high`, `wdg_irq_value_low` }. Then software should stop `wdg` so that `wdg` reset is not generated. If software cannot response to this interrupt, then `wdg` reset will be generated finally when `wdg_cnt` decrease to 0 which reboots the entire system.

When software completes using watchdog, it should disable `wdg_open` firstly, then it disables `wdg_rtc_eb` and `wdg_eb` in global control registers.

It's IMPORTANT that:

- Only writing `wdg_ld_value_high` cannot reload watchdog. Both `wdg_ld_value_low` and `wdg_ld_value_high` should be written if watchdog need reloaded. And software must write `wdg_ld_value_high` firstly, then write `wdg_ld_value_low`.
- There are 3-5 RTC clock cycles delay from setting `wdg_ld_value_low` to counter setting success because of different clock domains, just as timer does.
- It's not recommended to read counter value in normal usage, just as timer does.

9.14 VAD

9.14.1 Overview

VAD block is implemented into 2 parts: 1st step and 2nd step. The blocks of 1st step finish the pre-process of voice signal, and compare the energy of the signal with the threshold. If the average signal is over the threshold, the 2nd step will be active. The VAD core in 2nd step will give a decision that the input signal is human voice or not. Wakeup signal will be generated to CPU or DSP core, to indicate that human voice is detected. The data buffer in 2nd step will buffer the voice data for CPU or DSP core.

9.14.2 Features

- Support 2M input main clock
- The input 1bit data is 2M rate from PMIC
- Support PDM input data from DMIC
- Support HPF bypass by control register
- Support 16bits output data
- Support digital gain before output from HPF
- Support HBF function to down-sample 16K to 8K
- Support HBF bypass
- Support related threshold configurable in 1st step
- Support refresh timer window configurable in 1st step
- Support average timer window selected by control register in 1st step
- Support human voice detecting in 2nd step
- Data buffer is designed for data buffering during CPU or DSP waking up
- DMA function is supported for data transferring

9.14.3 Function Description

- CIC:output 80kHz signal with 16bits width
- DECI:Decimation Filter, output 16kHz signal with 16bits width
- HPF:High-pass Filter, output 16kHz signal with 16bits width
- HBF:Output 8K sample rate signal with 16bits width
- CMP: Generate the ABS of the input signal, and compare with the threshold. If the ABS value is bigger than the threshold, CMP Trigger will turn to 1, otherwise CMP trigger keeps 0. of The following comparing is supported:
 - Relative Threshold: If wakeup keeps 0, the average ABS of several samples will be calculated every period of time. The number of samples can be configured. The result of average ABS added with the threshold configured by DSP, is the real threshold. Each signal will be compared with the real threshold. If wakeup==1, the average ABS will not be updated.

GenEn:The second_enable signal (1 bit) is generated from the CMP trigger signal(1 bit) and the vad_flag signal(1 bit, from 2nd step). The logic is:

```
if CMP_trigger goes from 0 to 1 && second_enable == 0
    second_enable <= 1; //start
if vad_flag == 0 && second_enable == 1
    second_enable <= 0; //end
```

9.14.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.15 TDM

9.15.1 Overview

TDM is used to receive and transmit multi-channel audio data with external devices by two groups of IIS interface. It can transmit data by multi channels up to 8 slots. It can work as master or slave mode.

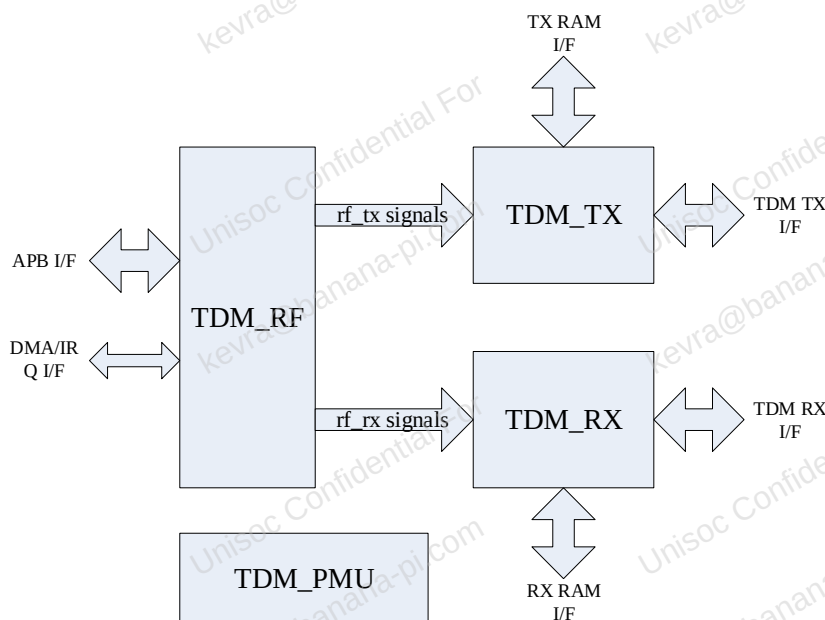
9.15.2 Features

- Support TX master, TX slave, RX master, RX slave modes
- Support full duplex or half duplex mode
- Support slot number is 2, 4, 8
- Programmable valid slot position
- IIS lrc mode supports sync 50% or pulse
- Support sample rate 48k/96k
- TDM can be configured as master or slave
- IIS data lengths is 16,24,32bits
- Slot width is 16/32 bits
- Support IIS compatible, left-justified(LJ), right-justified(RJ) data format
- Support MSB / LSB mode
- Detect overflow for RX and underflow for TX
- Use 256×32 TX FIFO and 256×32 RX FIFO
- Use high frequency clock to capture IIS input signals.
- Threshold control for TX/RX
- Support count control

9.15.3 Function Description

9.15.3.1 TDM Diagram

Figure 9-68 TDM Top Diagram



Software can read/write sram or configuration registers through APB interface.

Before transmitting serial data, software should configure external devices, then it writes TX sram data and configuration registers, enable TX controller at last.

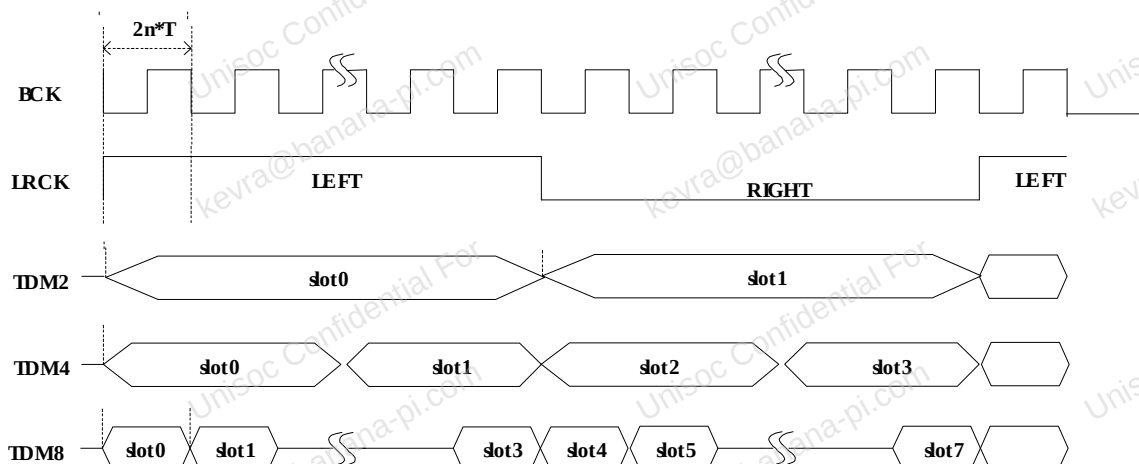
When TX controller is enabled and in master mode, it will read data from TX sram and generate TDM timing as required. When the left sram data number is less than Threshold, an interrupt signal will be generated to software or a DMA request signal will be generated to DMA controller. Software or DMA controller should write a fixed number of data to TX sram quickly. If TX sram is empty, TX controller will treat it as an underflow error, and send interrupt.

Before receiving serial data, software should configure external devices, and start external devices, then it writes TDM IP configuration registers, and enable RX controller.

When RX controller is enabled and in master mode, it will generate TDM timing as required, and capturing serial data using `clk_tdm_hf` at the positive edge of `bck`, transfer the captured serial data to `clk_tdm` domain, and collect to 32bit data. Then write the 32bit data to RX ram. When the left sram data number is larger than Threshold, an interrupt signal will be generated to software or a DMA request signal will be generated to DMA controller. Software or DMA controller should read a fixed number of data to RX sram quickly. If RX sram is full, RX controller will treat it as an overflow error, and send interrupt.

9.15.3.2 TDM Timing

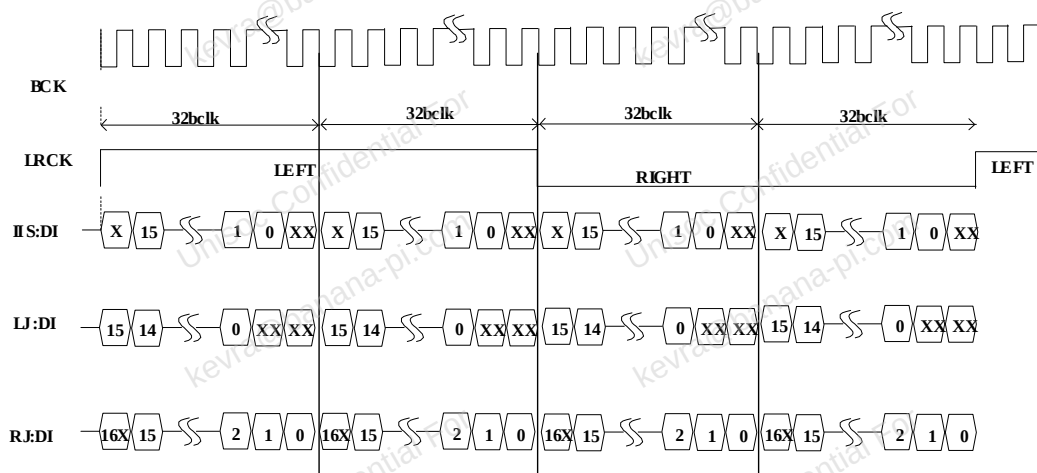
TDM interface slot number can be configured to 2/4/8.the sequence as below. Slot width can be configured to 16/32 bit

Figure 9-69 TDM basic timing


9.15.3.2.1 Tdm_4_Data_16bit_slot_length_32bit

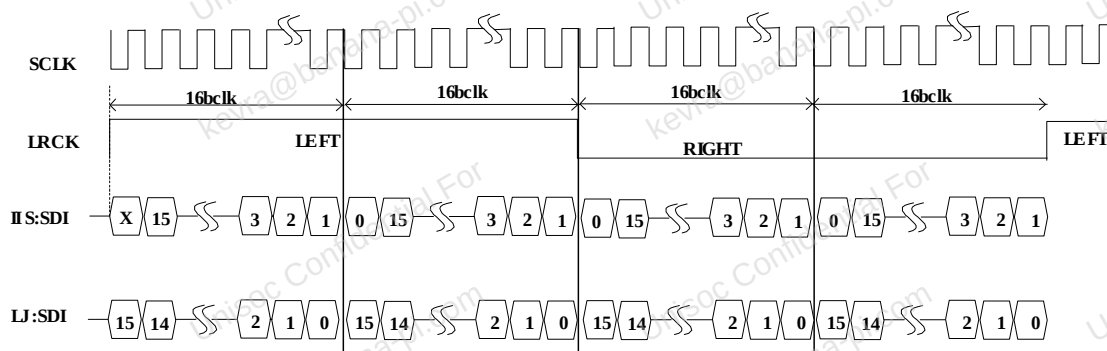
TDM slot num = 4. Sample rate = 48K, slot width = 32. Bit clk = $32 \times 4 \times 48k = 6.144M$

Audio IIS interface format supports IIS compatible, left justified, right justified.

Figure 9-70 TDM4 data16bit slot length32bit


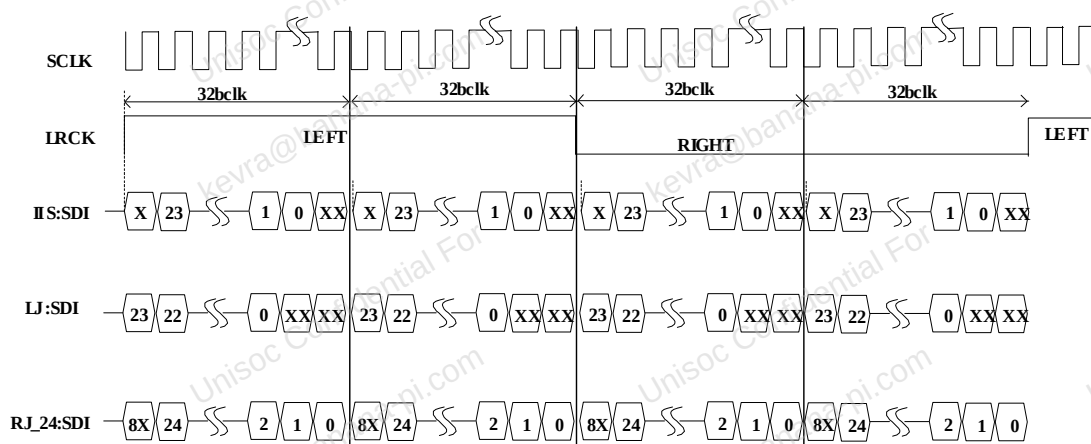
9.15.3.2.2 Tdm_4_Data_16bit_slot_length_16bit

Figure 9-71 TDM4 data16bit slot length16bit



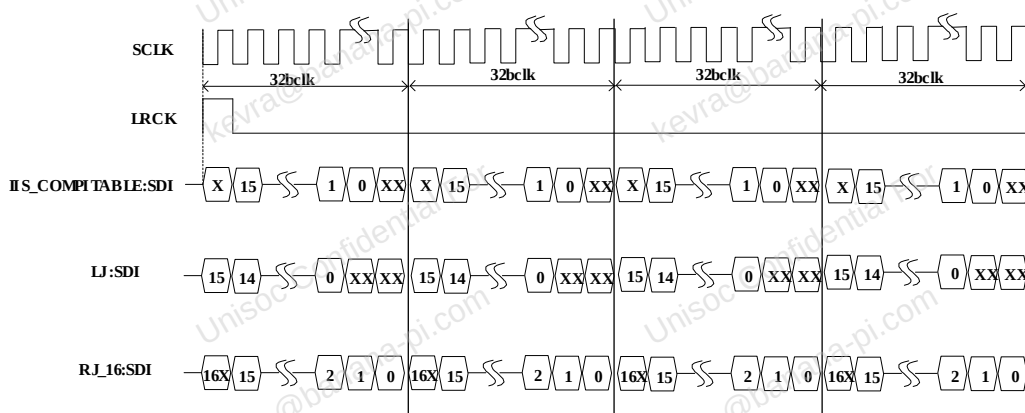
9.15.3.2.3 Tdm_4_Data_24bit_slot_length_32bit

Figure 9-72 TDM4 data24bit slot length32bit



9.15.3.2.4 Lrck mode

Figure 9-73 TDM4 data16bit slot length32bit



9.15.3.3 TDM BIT CLOCK

- Sample rate = 48k

Mode	Slot_width=16bit	Slot_width=32bit
TDM2	$2 \times 16 \times 48k = 1.536M$	$2 \times 32 \times 48k = 3.072M$
TDM4	$4 \times 16 \times 48k = 3.072M$	$4 \times 32 \times 48k = 6.144M$
TDM8	$8 \times 16 \times 48k = 6.144M$	$8 \times 32 \times 48k = 12.288M$

- Sample rate = 96k

Mode	Slot_width=16bit	Slot_width=32bit
TDM2	$2 \times 16 \times 96k = 3.072M$	$2 \times 32 \times 96k = 6.144M$
TDM4	$4 \times 16 \times 96k = 6.144M$	$4 \times 32 \times 96k = 12.288M$
TDM8	$8 \times 16 \times 96k = 12.288M$	$8 \times 32 \times 96k = 24.576M$

9.15.3.4 SRAM

There are one 256×32 TX sram and one 256×32 RX sram.

Each 32bit data represents only one audio data. Not support two 16bit audio data merge into one 32bit data, and write to TX sram, or read from RX sram and treats as two 16bit audio data.

TX sram and RX sram share the same APB address:

- Writing any address between 0x400 to 0x7FF will write data to TX sram.
- Reading any address between 0x400 to 0x7FF will read data from RX sram.

9.15.4 Control Registers

See *UIS7885 Audio System Control Registers Instructions*.

9.15.5 Application Notes

9.15.5.1 TX start flow

1. Configure TX_CFG0, TX_CFG1.
2. Software reset TX controller by writing 1 to tx_soft_rst bit in TX_CTRL and then writing 0 to it.
3. Fill TXFIFO by writing data to TXFIFO address, the number of data should be more than tx_threshold.
4. If TX is slave mode, set tx_io_en to 1, after 100 ns.
5. Set tx_enable to 1.
6. Wait tx_almost_empty interrupt, if this interrupt occurs, write TXFIFO with wcount number of data, $wcount + tx_threshold < 256$, $wcount > tx_threshold$.

9.15.5.2 TX suspend flow

Only valid in TX master mode and LJ or RJ mode.

After TX start, Wait tx_almost_empty interrupt. After this interrupt occurs, can wait any time before writing TXFIFO. During this time, tx_underflow interrupt should be ignored. tx_threshold should be less than 0x70 in suspend case

9.15.5.3 TX end flow

1. Clear tx_enable to 0, and wait tx_st in TX_STAT to 0.
2. Clear tx_io_en to 0 if TX is in slave mode.

9.15.5.4 RX start flow

1. Configure RX_CFG0, RX_CFG1
2. Software reset RX controller by writing 1 to rx_soft_rst bit in RX_CTRL and then writing 0 to it
3. Set rx_io_en to 1, after 100 ns
4. Set rx_enable to 1
5. Wait rx_almost_full interrupt, if this interrupt occurs, read RXFIFO with rcount number of data, $rcount = rx_threshold$

9.15.5.5 RX end flow

1. Clear rx_enable to 0, and wait rx_st in RX_STAT to 0.
2. Clear rx_io_en to 0.

9.15.5.6 Recommended Setting

1. TX and RX threshold to set to 128
2. Use TX_CFG2 and RX_CFG2 to specify how many data to be transferred, bit[31] should be set when use count control
3. The total latency (from req to ack) should be less than 128×600 ns when threshold is 128.

10

MultiMedia Subsystem

10.1 Camera

10.1.1 General Description

10.1.1.1 Overview

Camera system receives and process the image data from sensor, completes preview, capture, recording, and post processing. Camera system mainly contains DCAM, ISP and VDSP blocks.

The general features supported by the camera are as following:

- DCAM process the real-time online image data from the MIPI PHY, and has low-latency path to access the DDR memory
- ISP is the post-processing unit for the image data, and has high-bandwidth channel to access the DDR memory
- VDSP is the Cadence VQ7 DSP, can be used for image processing and vision computing, and also as an alternative accelerator for AI
- Support JPEG codec
- Support DVFS
- Support security

With the powerful hardware engine and rich set software package, the system can also support:

- Support dual PDAF: Single PDAF type 1/2/3, Dual PD mode 3/4
- Support TOF
- Support super low light capturing for still and video
- Support auto scene recognition like: Portrait, Children, Backlight & portrait, portrait, Low light, Document, Macro, Standard, Landscape, Black and white, Spotlight, Sports
- Support HDR
- Support 3A (Auto Focus, Auto White Balance, Auto Exposure) and anti-flicker functions
- Support multi camera fusion: depth map, real-time bokeh, refocus, real-time optical-like zoom, and T + W capture fusion

The detailed feature of the hardware blocks will be described in the following paragraphs.

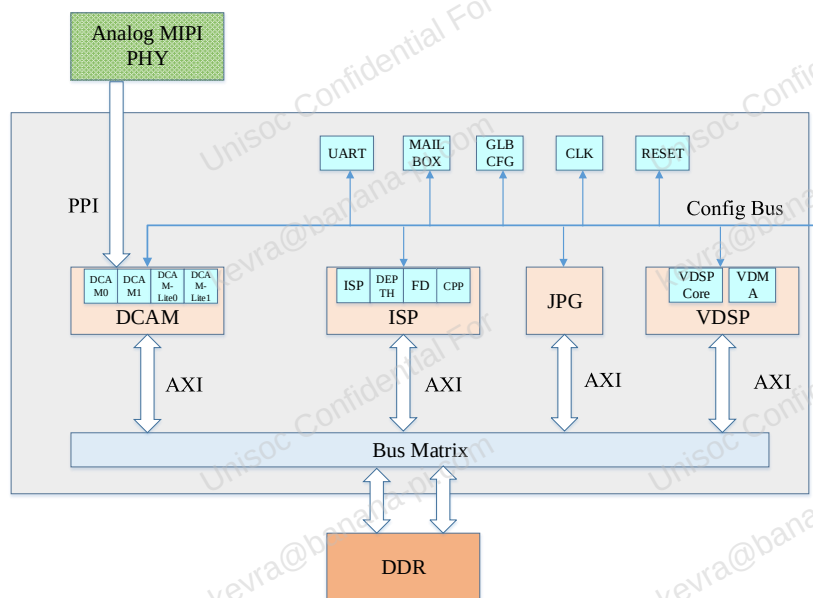
Numbers of sensors and application scenarios that the Camera systems supported are listed in the following table.

Table 10-1 Cam Application Case

Single Sensor	<ul style="list-style-type: none"> • 30fps preview (1440 × 1080, 1920 × 1080, 1920 × 1200, 2160 × 1080, 2560 × 1080, 3360 × 1440) • 64M(4in1)@25fps ZSL capture(full + preview) • 108M(9in1) @10fps non ZSL capture(full + preview) • Burst shot up to 48M10fps jpeg • Up to 64M RAW + jpeg ZSL capture • Slow motion up to 4k @60fps, 1080p@240fps
Dual Sensors	<ul style="list-style-type: none"> • Up to 24M + 16M • Up to 20M + 20M • Up to 32M 25fps + 8M 25fps
Triple Sensors	<ul style="list-style-type: none"> • Up to 108M(9in1) + 20M(UW) + 13M(T) • Up to 64M(4in1) + 20M(UW) + 13M(T)
Four Sensors	<ul style="list-style-type: none"> • Up to 108M(9in1) + 20M(UW) + 13M(T) + 2M(bokeh/video/Micro) • Up to 64M(4in1) + 20M(UW) + 13M(T) + 2M(bokeh/video/Micro)

10.1.1.2 Block Diagram

The block diagram of the Camera System is drawn in the figure above.

Figure 10-1 Camera System Block Diagram


10.1.2 DCAM Block

10.1.2.1 Overview

DCAM process the real-time online image data from the MIPI PHY, and has low-latency path to access the DDR memory. The MIPI PPI data comes from the embedded CSI MIPI PHY and feeds into the DCAM block.

10.1.2.2 Features

Features supported by the DCAM:

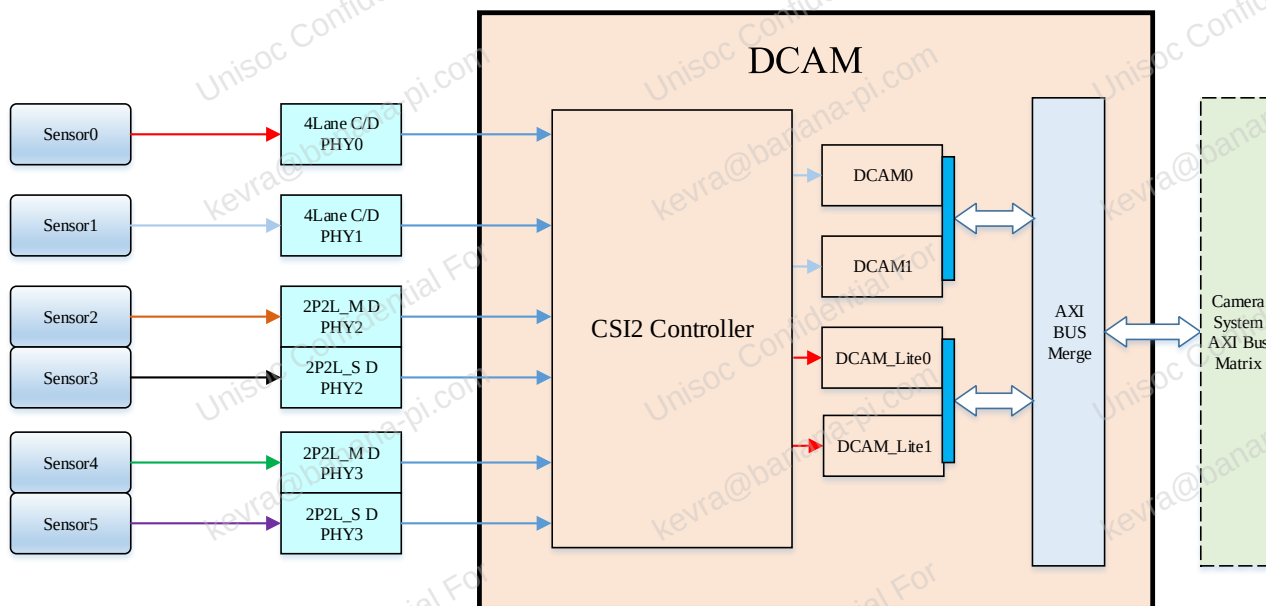
1. Two full-feature DCAM modules(DCAM0 and DCAM1) which support:
 - Support pre-binning for 4in1 sensor or normal sensor with bypass in RAW domain image data process (only DCAM0 and DCAM1).
 - Support Black Level Correction, RGB Gain control, PDAF Pixel Extraction and Correction, Lens Shading Correction, Crop, Auto White Balance Correction, Bad Pixel Correction, 3DNR_ME, NLM, Imbalance, GrGb and CFA with bypass in Raw domain image data process.
 - Support CCM, Global Tone Mapping, Gamma10 and CCE with bypass in Full RGB domain image data process.
 - Support YUV422to420 and YUV Scaler0 with bypass in YUV domain image data process.
 - Support cropped full size raw data capture with or without pre-process in Full path.
 - Support YUV Scaler or Bypass in Preview path.
 - Support Pyd_dec1/2/3/4 or Bypass in Preview path.
 - Support FBC encoder in RAW path, Capture path and in Preview path.
 - Support type1/2/3 PDAF info, mode 3/4 dual PD and some other info capture via data type mode or virtual channel mode in all VCH path;
 - Support original RAW data output in VCH2 path together with pre-processed Raw data output in Raw path concurrently at debug mode.
 - All data paths work with AXI write access
 - Support PDAF Pixel Extraction with AXI write
 - Support configurable 32×32 to 128×128 block in LSCM with AXI write
 - Support Lens Shading Correction with AXI read
 - Support configurable 32×32 to 128×128 block in Auto Exposure Monitor module with AXI write;
 - Support Bayer Histogram with AXI write;
 - Support Bayer2Y + Anti-Flicker with AXI write;
 - Support Bad Pixel Correction with both AXI write and AXI read;
 - Support configurable 4×3 to 20×15 block in Auto Focus Monitor module with AXI write;
 - Support 3D Noise Reduction Motion Estimation with AXI write;
 - Support image data fetch from DDR for offline raw domain process in Preview path (only DCAM1)
 - Support IOMMU and AFBC
 - Support Timestamp
2. Two lite DCAM modules (DCAM-Lite0 and DCAM-Lite1) which support:
 - Support horizontal and vertical decimation for YUV data;
 - Support RAW RGB from MIPI, 8bits, 10bits and 12bits;
 - Support YCbCr from MIPI, 8bits and 10bits
 - All data path work with AXI write access

- Support IOMMU
- Support Timestamp
- 3. Four CSI MIPI Controllers that support:
 - MIPI® Alliance Specification for Camera Serial Interface 2(CSI-2) Version 1.3 r06 - 22Jan. 2014
 - MIPI® Alliance Specification for D-PHY v1.2 - 14September 2014
 - MIPI® Alliance Specification for C-PHY v1.0
 - Support both C-PHY and D-PHY
 - Dynamically configurable multi-lane merging
 - Error detection and correction at PHY level, Packet level, Line level and Frame level
 - Long and Short packet decoding
 - Image Pattern Generator is added for in BIST mode
 - Several Frame formats
 - General Frame or Digital Interlaced Video with or without accurate sync timing
 - Data type(packet or frame level) and virtual channel interleaving
 - All primary and secondary data formats
 - RGB, YUV, and RAW color space definitions
 - From 24-bit down to 6-bit per pixel
 - Generic or user-defined Byte-base data types
 - Support 2×2 lane and 4 lane mode
 - Support ipg mode virtual channel loop
 - Support ipg mode up to 108M for built-in test.

10.1.2.3 Function Description

The DCAM block diagram is in the block labeled as “DCAM” in [Figure 10-2](#). The sensor and MIPI PHY interface also shown in the figure. The MIPI PHY is implemented in the chip also. The MIPI PHY convert the serial analog sensor signals to PPI data and output to the CSI2 controller. Up to six sensors can be connected to the chip and only four can be processed simultaneously. Then four out of six outputs from the CSI controller are selected and routed to the four online processing modules: DCAM0/1 and DCAM-Lite0/1.

Figure 10-2 DCAM Block Diagram



There are four analog MIPI PHYs implemented in the chip. The feature of the PHYs are described in the following table.

Table 10-2 MIPI PHY Description

PHY NAME	MIPI PHY compliance	Mode of Operation	Description
C/D PHY0 (combo0)	<ul style="list-style-type: none"> MIPI C-PHY 1.2, 3.5Gsp/s MIPI D-PHY 1.2, 2.5Gbps 	<ul style="list-style-type: none"> C-PHY, 3 trio D-PHY <ul style="list-style-type: none"> 4 lanes. 2 lanes: lane0 and lane1 	Combo PHY
C/D PHY1 (combo1)	<ul style="list-style-type: none"> MIPI C-PHY V1.2, 2.5Gsp/s MIPI D-PHY 1.2, 2.5Gbps 	<ul style="list-style-type: none"> C-PHY, 3 trio D-PHY <ul style="list-style-type: none"> 4 lanes. 2 lanes: lane0 and lane1 	Combo PHY
2P2L DPHY 2	MIPI D-PHY 1.2, 2.5Gbps	<ul style="list-style-type: none"> One port 4 lanes Two ports, 2 lanes each. Connect to 2 sensors 	DPHY only
2P2L DPHY 3	MIPI D-PHY 1.2, 2.5Gbps	<ul style="list-style-type: none"> One port 4 lanes Two ports, 2 lanes each. Connect to 2 sensors 	DPHY only

There are four CSI2 controller in the block. The signal connections between the PHY and CSI2 controller are fixed.

The CSI2 controller conforms to the following standard.

- MIPI® Alliance Specification for Camera Serial Interface 2(CSI-2) Version 1.3 r06 - 22Jan. 2014
- MIPI® Alliance Specification for D-PHY v1.2 - 14September 2014
- MIPI® Alliance Specification for C-PHY v1.0

Both CSI2_controller0 and CSI2_controller1 can be configured to be adapt to either C-PHY or D-PHY. The number of data lanes can also be configured through software.

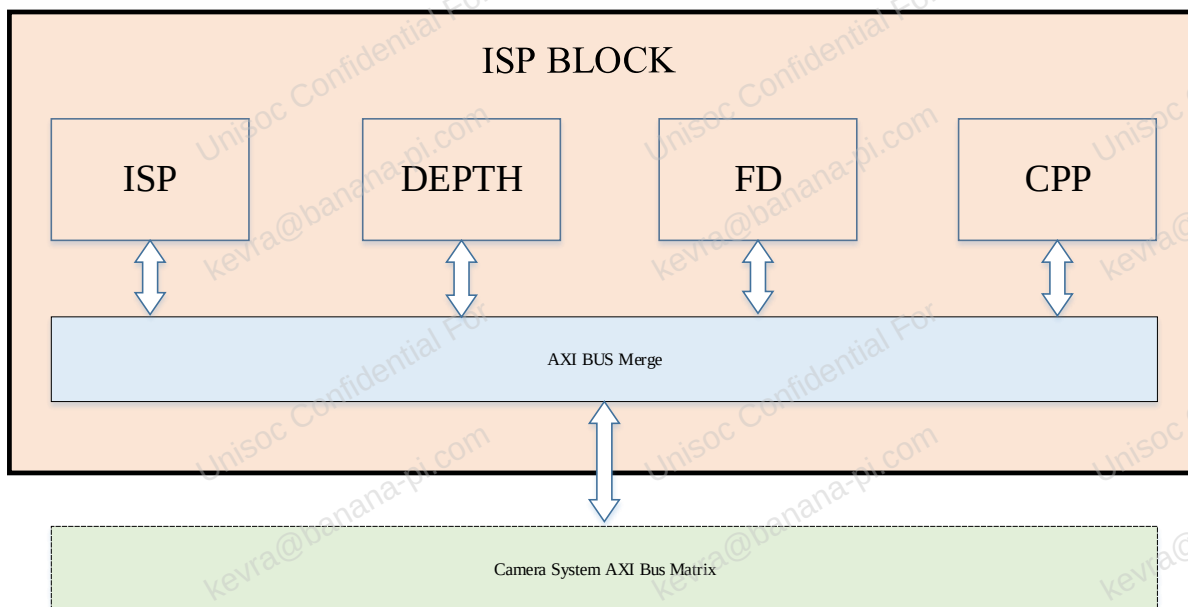
CSI2_controller2 and CSI_controller3 can be configured to work in two modes: one port with 4 lanes; or two ports with 2 lanes each. When working in one port 4 lanes mode, the output 32-bit MIPI data will be output on port1 of the controller.

The working mode configuration of the CSI2 controller must match the MIPI PHY working mode.

10.1.3 ISP Block

The ISP block contains four modules ISP, DEPTH, FD and CPP for offline image processing. The data for processing is read from or written into the DDR memory.

Figure 10-3 ISP Block Diagram



10.1.3.1 ISP

10.1.3.1.1 Overview

ISP can handle two kinds of input data format: YUV 420 (8-bit, 10-bit) and Full RGB (10-bit). ISP fetches YUV 420 data from DDR and outputs YUV 420 and YUV 400 image after a series of image processing.

10.1.3.1.2 Features

The ISP Support:

- Full RGB space
 - LTM (Local Tone Mapping)
 - HSV(upgrade V thresholding)

- 3D_LUT
- RGB to YUV420 conversion
- YUV space
 - UVD (UV division)
 - 3DNR
 - CDN (Color Denoise)
 - YNR
 - CNR
 - YUV420 to RGB conversion
 - Y-Gamma
 - Edge Enhancement (with pyramid Gaussian filter)
 - Y-random
 - BCHS(Brightness, Contrast, Hue, Saturation)
 - YUV Scaler1 (preview/capture)
 - YUV Scaler2 (recording)
 - YUV Scaler3 (FD/Thumbnail, video format output for face detection)
 - Noise filter
 - YUV Regular2
 - YUV Regular3
 - Dewarping
 - Pyd_rec
 - CNR
- Dec Offline capture space
 - AFBD
 - Dec_bwu (Decompose bit width upper)
 - DCT
 - Pyd_dec(Pyramid Decompose) offline
 - Dec bwd
- FBC/FBD for reducing DRAM bandwidth of YUV420 frame data access
- Support IOMMU
- Low Power control
- ISP timeout interrupt
- Security control
- Support AFBC
- 2 pixel/cycle capacity
- Support Timestamp

10.1.3.1.3 Function Description

ISP works as an independent module to focus on the image processing and cooperates with Digital Camera Interface (DCAM), video encoder, LCD Controller (LCDC), GPU or vision processor. DCAM receives the input pixel data from Camera Sensor. The received data will be sent to the external memory after processing. The image data in external memory then are loaded into the ISP pipeline and continue the subsequent processing. *ISP Data Fetch* sub-module is responsible for data fetching from the external memory. The received YUV data then are

handled in *YUV* Pre-sub-module, the *Full RGB* sub-module and *YUV* Post-sub-module successively. The received data can also be *Full RGB*, but this mode is only used for verification. Dewarping and *pyd_reconstruction* modules are also added to the ISP system, input through different data channels, and then output through a data channel. Finally, the results will be stored to the external memory by the Store sub-module. LCDC can display these processed data by the software.

Pyramids Deconstruction is separate pipeline, and it can work with *fmcu_dec*. It just share the AXI read/write ports with ISP.

10.1.3.2 CPP

10.1.3.2.1 Overview

The CPP is a hardware engine for generating the thumbnails, supports scaling and rotation.

10.1.3.2.2 Features

The CPP supports the following features:

- Path0 can support one input with two output (scaler output and bypass output) feature
- Path0 scaler can support initial phase configuration
- Support separate rotation path, which is path1
- Support separate DMA path
- Support scaling factor $1/64 \sim 8x$
- Maximum input size $8K \times 8K$
- Support rotation on 64×64 block

10.1.3.2.3 Function Description

CPP can support three separate independent path:

- Path0, which is scaler + bypass path, (bypass path can be disabled).
- Path1, which is rotation path.
- DMA path.

10.1.3.2.4 Control Registers

See *UIS7885 Camera System Control Registers Instructions*.

10.1.3.3 FD

10.1.3.3.1 Overview

The FD (face detection) module supports still mode and movie mode, and the face model is configurable

10.1.3.3.2 Features

The FD supports the following feature:

- Still mode & movie mode.
- Image size up to 640×480 .
- Minimum Face Size is 20×20 .
- Up to 8191 face detection.
- Up to 128 area to face detection in movie mode.

- Model information can be configured by software.

10.1.3.3.3 Function Description

Face detection data flow is as shown below, which calculates multi-dimension image according to the original image; calculates feature images on each dimension image; perform slide detection window on each dimension feature image with window size equal to 24×24 pixels; uses multi-stage face detection classifier to distinguish face or non-face in each sliding window.

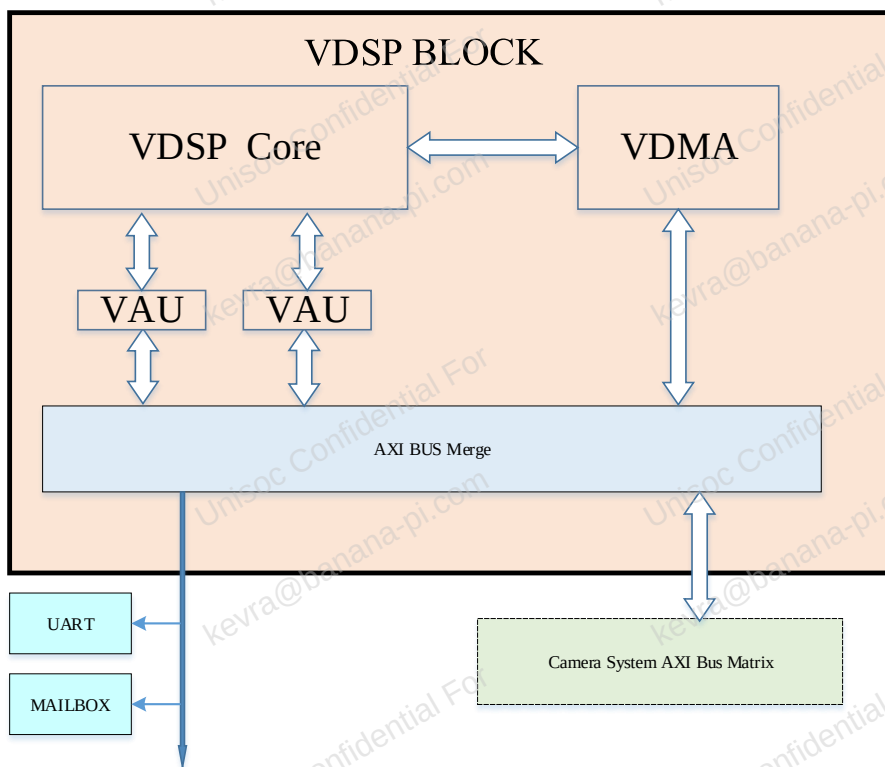
10.1.3.3.4 Control Registers

See *UIS7885 Camera System Control Registers Instructions*.

10.1.4 VDSP Block

The VDSP block integrates the Tensilica VQ7 DSP core, VDMA, MMU, UART and Mailbox.

Figure 10-4 VDSP Block Diagram



10.1.4.1 VDSP Core

10.1.4.1.1 Overview

The Cadence Tensilica VQ7 DSP IP core. For image processing and vision computing. Also as an AI accelerator, supported by the Tensilica Neural Network Compiler, it can map any neural network trained with a framework such as Caffe, TensorFlow, and TensorFlow Lite into executable and highly optimized fixed-point code

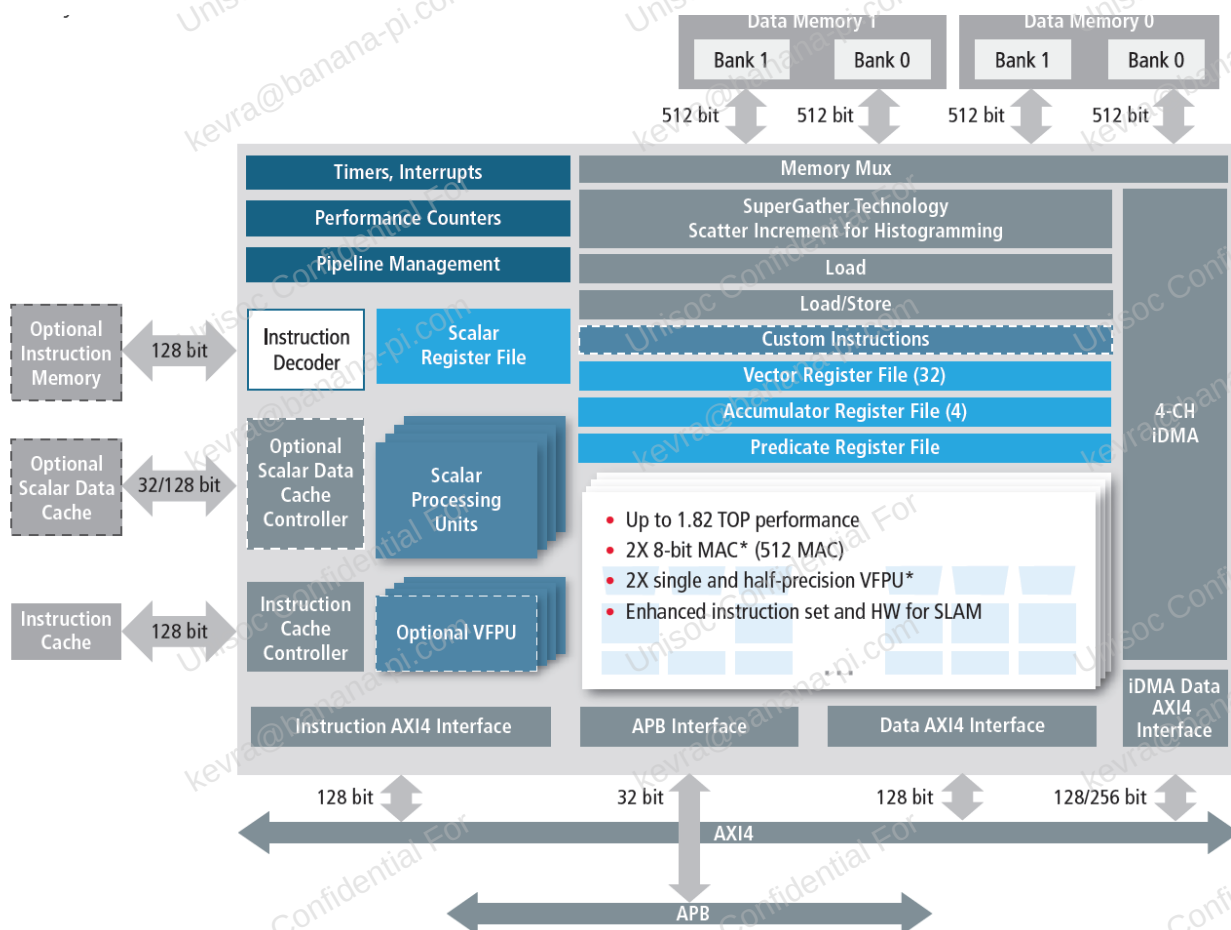
10.1.4.1.2 Features

The VDSP supports the following feature:

- Provides high-performance, energy-efficient vision and AI embedded DSPs
- Up to 1GHz working clock
- Very long instruction word (VLIW) SIMD architecture delivers up to 1.7X higher TOPS compared to the Vision Q6 DSP in the same area.
- An enhanced instruction set supporting 8/16/32-bit data types and optional VFPU support for single and half precision enable up to 2X faster performance on SLAM kernels compared to the Vision Q6 DSP
- Delivers up to 2X improvement in floating-point operations per mm2 (FLOPS/mm2) for both half-precision (FP16) and single precision (FP32) compared to the Vision Q6 DSP
- Up to 2X greater AI performance in the same area compared to the Vision Q6 DSP results in up to 2X improvement in GMAC/mm2 compared to the Vision Q6 DSP
- Optimized for simultaneous localization and mapping (SLAM)
- Provides a complete subsystem using the integrated DMA that allows the system to transfer high-resolution data directly into the local memory of the DSP, thus hiding the data access latency associated with accessing data from an external DRAM
- Implements the Tensilica SuperGather™ enhanced memory interface to quickly and efficiently read/write non-contiguous locations from local memory
- Delivers scalable, optimized performance with low energy for computer-vision and pixel-processing applications that span a large range of data types from 8b to 32b, such as face detection, object detection, lens distortion correction, and many more advanced vision applications

10.1.4.1.3 Function Description

Figure 10-5 VDSP Core Block Diagram



The Vision DSP architecture incorporates advanced VLIW/SIMD support for the industry's highest number of ALU and MAC operations per processor cycle, as well as the industry's widest and most flexible memory bus.

Specialized instructions also allow the Vision DSP to efficiently speed up pixel processing. Various architecture enhancements boost the performance while keeping the energy consumption low. The Vision DSP family provides unprecedented flexibility in system implementations at power-consumption levels that significantly reduce the need for hardware accelerators. The DSPs also offer an integrated DMA engine, interface for instruction memory, instruction cache, and two AXI interfaces.

10.1.4.1.4 Control Registers

See *UIS7885 Camera System Control Registers Instructions*.

10.1.4.2 VDMA

10.1.4.2.1 Overview

VDMA is a specified DMA for VDSP, which includes data copy, crop, AFBCE, AFBCD, unpack2pack, pack2unpack, and padding function

10.1.4.2.2 Features

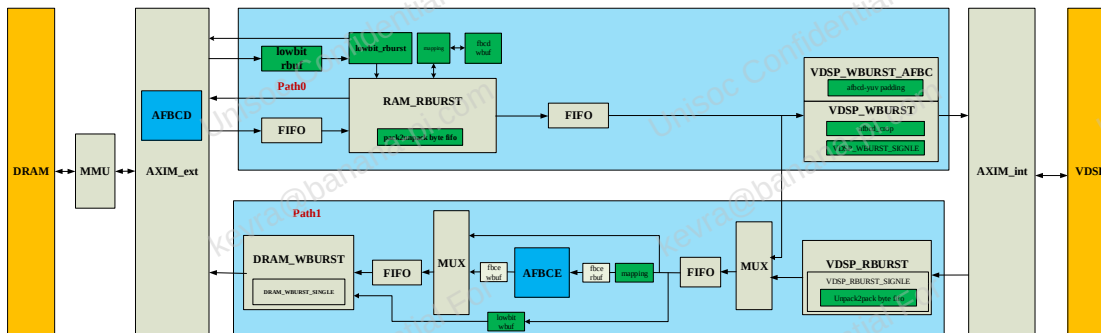
The VDMA supports the following feature:

- Support two AXI bus. One is for DRAM, another for VDSP.
- Support two separated DMA path, which can works simultaneously.
- Support DRAM to DRAM DMA channel, when the VDSP DMA channel is idle.
- Both DRAM read and write channel can support these data formats: AFBC-YUV/Y/UV, AFBC-RAW8/10/12/14, Y, UV ,pack-RAW10/12/14,unpack-RAW10.
- Both VDSP read and write channel can support these data formats: YUV, Y, UV, unpack-RAW8/10/12/14.
- Support block data access. Block size is configurable.
- Support minimum block 32×2 .
- Support command queue work mode.
- Support AFBCE/AFBCD-Y/UV.
- Support AFBCE/AFBCD-RAW8/10/12/14, support RGGB/GRBG bayer mode.
- Support unpack2pack/pack2unpack-RAW12/14.
- Support AFBCD-YUV/Y/UV 4 direction respectively padding.
- Support maximum image size 12032×9024 .
- Support embedded MMU.

10.1.4.2.3 Function Description

The VDMA can move image data from VDSP TCM to external memory, or vice versa. Also, it can move data from once location to another location in the external memory.

Figure 10-6 VDMA module Diagram



10.1.4.2.4 Control Registers

See *UIS7885 Camera System Control Registers Instructions*.

10.1.4.3 VAU

10.1.4.3.1 Overview

The VAU translates the virtual address sent from the VDSP core to the physical address. It maps contiguous virtual memory pages to scattered physical ones and support secure protection.

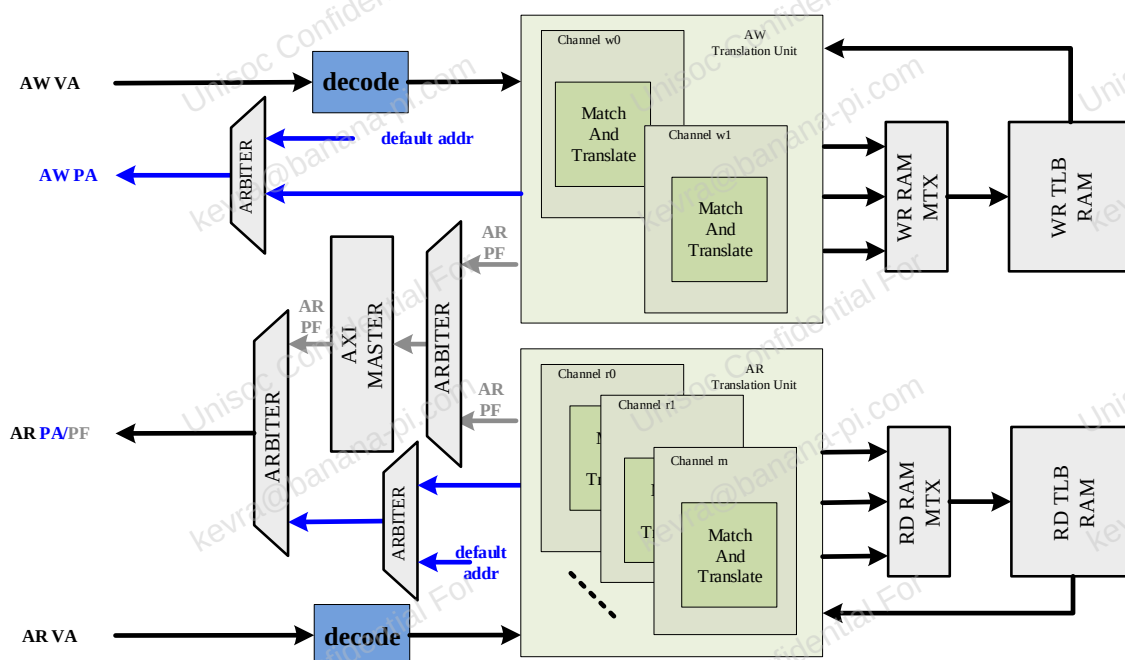
10.1.4.3.2 Features

The MMU support the following feature:

- One level RAM cache
- Has interrupts for error event
- RTL configurable pagetable entry: 4~16bit
- AXI 3/4 interface with User signals
- 4KB page size
- SW configurable address range
- One optional protection bit for read/write
- Three cycle delay for hitting of one level page table
- Performance monitor for a single channel
- Debug interface (AXI; FSM state;CHN State)

10.1.4.3.3 Function Description

Figure 10-7 VAU module Diagram



A TLB buffer stores some pagetable entries. If VA (virtual address) hits the entries in TLB buffer, VAU read the TLB RAM, the PPN and VA offset are combined into output PA. If not, VAU will fetch the pagetable entries in the DDR through AXI bus and fill them into TLB RAM. A FIFO is implemented for read channel to improve the efficiency of translation.

10.1.4.3.4 Control Registers

See *UIS7885 Camera System Control Registers Instructions*.

10.1.4.4 UART

10.1.4.4.1 Overview

The UART module serves as the log and debug port for the VDSP. The UART module includes a baudrate generator with software-programmable divider ratios for all common baudrates.

Two types deep FIFO are include. Receive FIFO which is 128-Byte-deep and send FIFO which can be configured to 128-Byte-deep or 256-Byte-deep;The module also includes a flexible interrupt with multiple maskable interrupt sources. Two hardware flow control lines are included (one for input, the other for output).

10.1.4.4.2 Features

The UART supports the following feature:

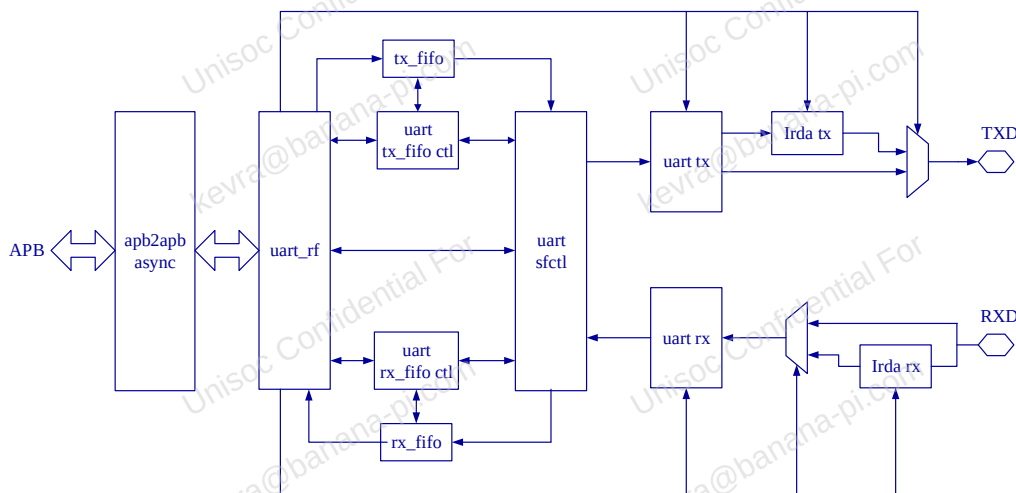
- Full-duplex operation
- Hardware flow control support
- Software flow control support with configurable command and escape register
- Configure TX FIFOs minimizes 128-Byte-deep or 256-Byte-deep processor overhead at high data rates
- Wide selection of UART word lengths, including 5,6,7 and 8 bits
- Stop bit number can be 1,1.5 or 2 bits
- RX stop bit number and TX stop bit number can be configured separately
- Automatic baudrate detection flow support
- Support odd/even parity
- Auto detect for parity and framing error
- UART loop-back test mode
- Break character detection and generation
- support DMA operation
- Single interrupt line for multiple maskable interrupt source events
- Programmable interrupt trigger levels for FIFOs
- Baudrate generation based upon programmable divisors, operating from a flexible functional clock
- Includes a built-in IrDA controller
- Support 32 bits or 8 bits write once to txfifo

10.1.4.4.3 Function Description

The UART and IRDA baudrate generator is based on the clk_uart, the divisor coefficient is controlled by register UART_CKD0. IRDA part clock period is the 1/16 of the UART part since the required timing of IRDA is 3/16 or 4/16 of bit width. The serial data rate can be changed by modify the clk_uart or the UART divisor coefficient.

The UART module has separated RX and TX channel to realize the duplex working mode.

Figure 10-8 UART module Diagram



10.1.4.4.4 Control Registers

See *UIS7885 Camera System Control Registers Instructions*.

10.1.4.5 MAILBOX

10.1.4.5.1 Overview

The MAILBOX module is used for the communication between VDSP core and other CPU cores in the SOC. Mailbox's main receive mails contain some control information from different processors, and send those mails to the right target, it will send proper IRQ to target/source processors to notice them the status about mails.

10.1.4.5.2 Features

The MAILBOX supports the following feature:

- Source processor write mail into InBox.
- Mail Controller delivers the mail into Target processor Box
- Box number depends on parameter(1~16)
- OutBox store mails in FIFO(sram)
- Each OutBox's FIFO depth can be set by register file independently(1~256)
- Debug mode supported, able to access all mail by debug pointer
- Two working mode: FIFO mode, IRQ mode
- Target OutBox sends out interrupt to Target processor if FIFO is not empty
- Source InBox sends out interrupt to Source processor if mail is send, block or overflow

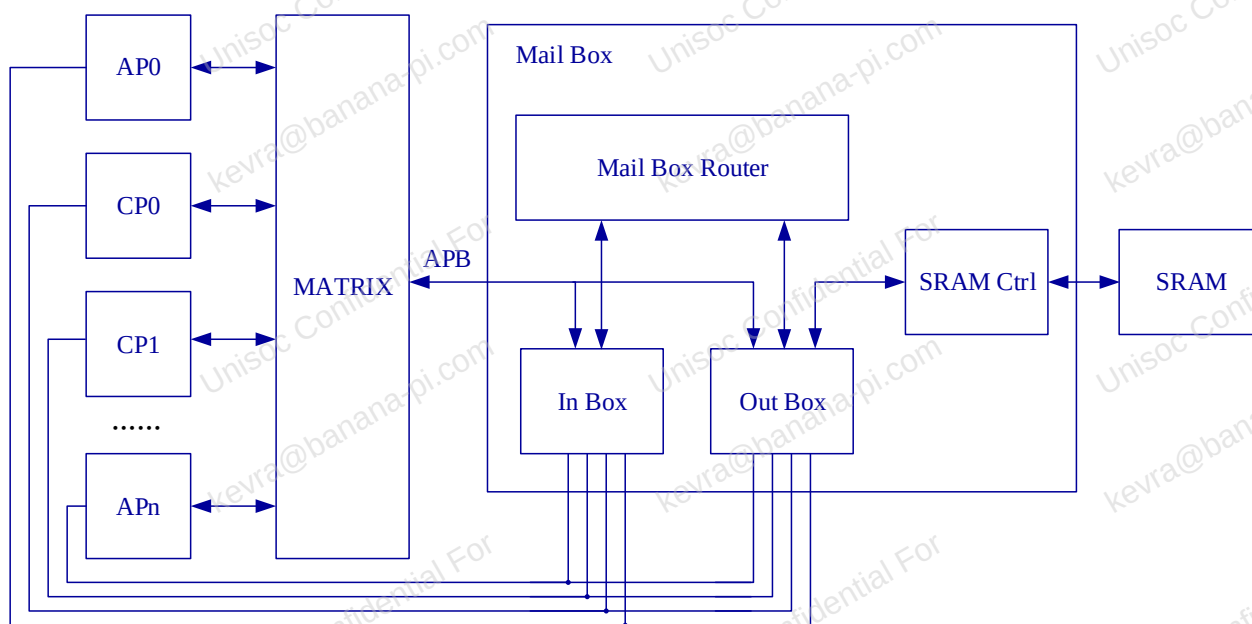
10.1.4.5.3 Function Description

Inbox has one 1-depth FIFOs for each outbox. Before source writes into it, source should set the target id register and lock the In Box. After the writing finished, source should release the Inbox. If Inbox is not empty, it will note the Router to get mail from it. Router will get one mail one time. The mail in inbox consists of target ID & mail content. If one mail is gotten by router, In Box could send an interrupt to Source. The mail in InBox could be read by other processor via the debug address. However, the read point will not be changed.

Out box will receive the mail coming from router, store mails to the FIFO(FIFO is not in outbox, out box will produce the read/write pointer and sram controller will provide the read/write signal), and notice the target processor to get it's mail. Mails are saved in sram, each outbox has configurable depth FIFO(1~256), all FIFO share one sram. The depth of each OutBox should be set by register file while module initialization, register reset values are set by parameter(FIFO_X_DEFAULT_DEPTH).

If Out Box is not full, it will ack the inbox's req immediately, and router will send mail to it. Router will send one mail one time, if multiple mails need to be send, inbox0 has the highest priority. If FIFO is not empty, it will initial the Interrupt to target processor to get mail from it. The mail in OutBox's FIFO could be read not follow FIFO order via the debug mode pointer. However, the FIFO read point will not be changed.

Figure 10-9 MAILBOX module Diagram



10.1.4.5.4 Control Registers

See *UIS7885 Camera System Control Registers Instructions*.

10.1.5 JPG

10.1.5.1 Overview

The JPG Module performs the compression and decompression function of the image.

10.1.5.2 Features

Features supported by the JPG:

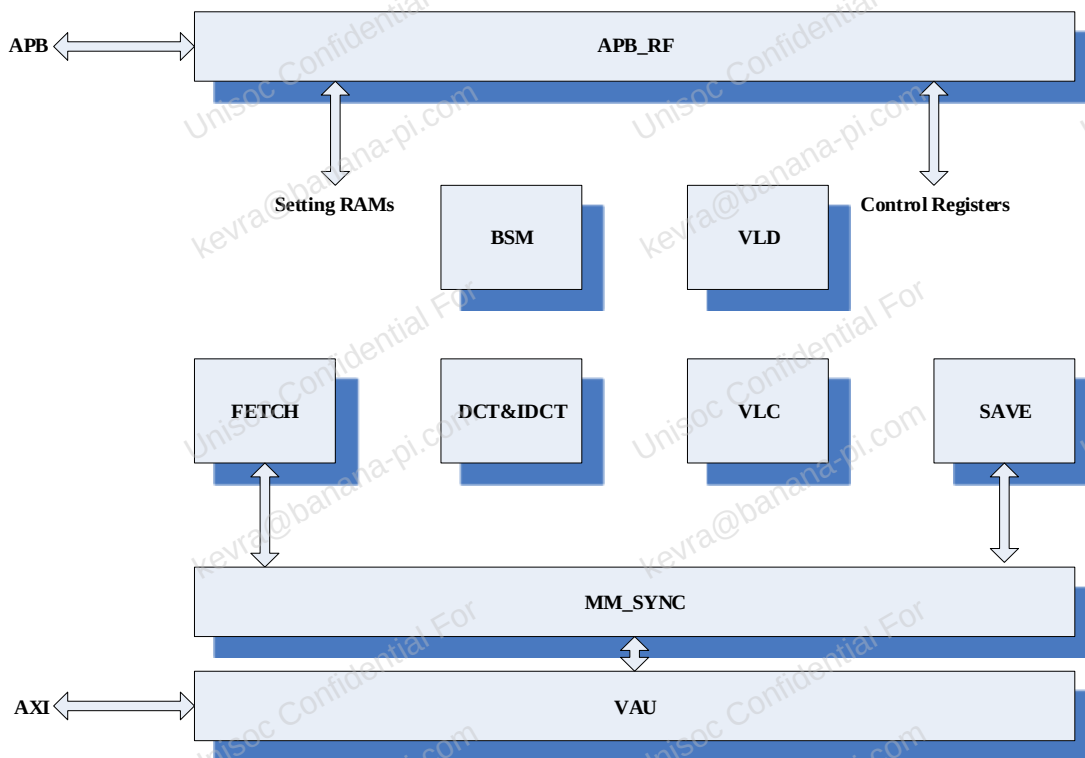
- Support jpg baseline profile encode and decode(size up to 256M)
- Support multi data format in encode:
 - YUV400,YUV420 and YUV422
 - YUV420 supports 2-plane and 3-plane, YUV422 only supports 2-plane
 - Y and UV pitch is configured respectively
 - If 2-plane, UV order is adjustable

- Support encode input rotation(90/mirror/mirror + 90/flip/flip + 90/180/270)
- Support multi data format in decode: include YUV444, YUV422, YUV420, YUV400, YUV411R, YUV422R, YUV411, all the formats are 2-plane(UV order is adjustable) and Y/UV pitch are the same
- Support output down sample in decode, scale is 1/2 or 1/4, down sample begins after all the formats are converted to YUV420 or YUV422
- Support YUV2RGB out in decode mode
- Support padding in horizontal and vertical direction in both normal and rotation encode mode
- Y/U/V base address, bit stream address in encode mode should be 8 Byte align
- Pitch in encode mode and decode mode(YUV or RGB) should be 4Bytes align
- Flexible dynamic ARQOS adjust mechanism
- Support MMU

10.1.5.3 Function Description

The JPG module diagram is shown as below.

Figure 10-10 JPG module Diagram



The function of the sub module is described as follows:

- **APB_RF**: configure the control registers and SRAM using APB4.0 protocol.
- **FETCH**: fetch YUV data in encode mode and prepare the MCU data for DCT; fetch bit stream for BSM in decode mode.
- **BSM**: accumulate the encoded data into 64-bit width data and stuff 0x00 after 0xff in the bit stream; in decode mode it disassemble the 64-bit bit stream according to the request from VLD and dyestuff the 0x00 after 0xff in the bit stream.

- **DCT&IDCT:** convert the YUV data using Discrete Cosine Transform or convert the frequency domain coefficient using inverse Discrete Cosine Transform.
- **VLD:** decode the non-zero coefficient according to Huffman Table.
- **VLC:** encode the non-zero coefficient according to Huffman Table.
- **SAVE:** transform the decoded YUV data format and execute down sample/yuv2rgb in decode mode or save the bit stream and deliver it out in encode mode.

10.1.5.4 Control Registers

See *UIS7885 Camera System Control Registers Instructions*.

10.1.6 Interface

This chapter describes the interface between the sensor and camera system.

MIPI

As described in the [10.1.2 DCAM Block](#), there are four analog MIPI PHYs implemented in the chip: two combo phys that support both C-PHY and D-PHY, and two 2p2lane phys that support D-PHY only.

10.1.6.1 Combo PHY

The Combo PHY supports both MIPI C-PHY V1.2 and MIPI D-PHY 1.2 protocol, and there are three data lanes(Trio) and implements the clock and data recovery inside in C-PHY mode; and four data lanes and one clock lane in D-PHY mode. The lane modules have the followed functions: HS-RX, LP-RX, and ULP-RX function.

The combo PHY support the following feature:

- TSMC7FF PLUS process
- $1.8V \pm 10\%$ and $1.2V \pm 10\%$ Analog supply operation
- $0.65V \sim 0.75V$ Digital supply operation
- Operation junction temperature range: $-40^{\circ}C \sim 125^{\circ}C$
- Lane operation ranging from 400 to 3500Mps in C-PHY HS mode
- Lane operation ranging from 80 to 2500Mbps in D-PHY HS mode
- Data rate in low-power mode: 10Mbps

The PAD of the interface is listed in the table below.

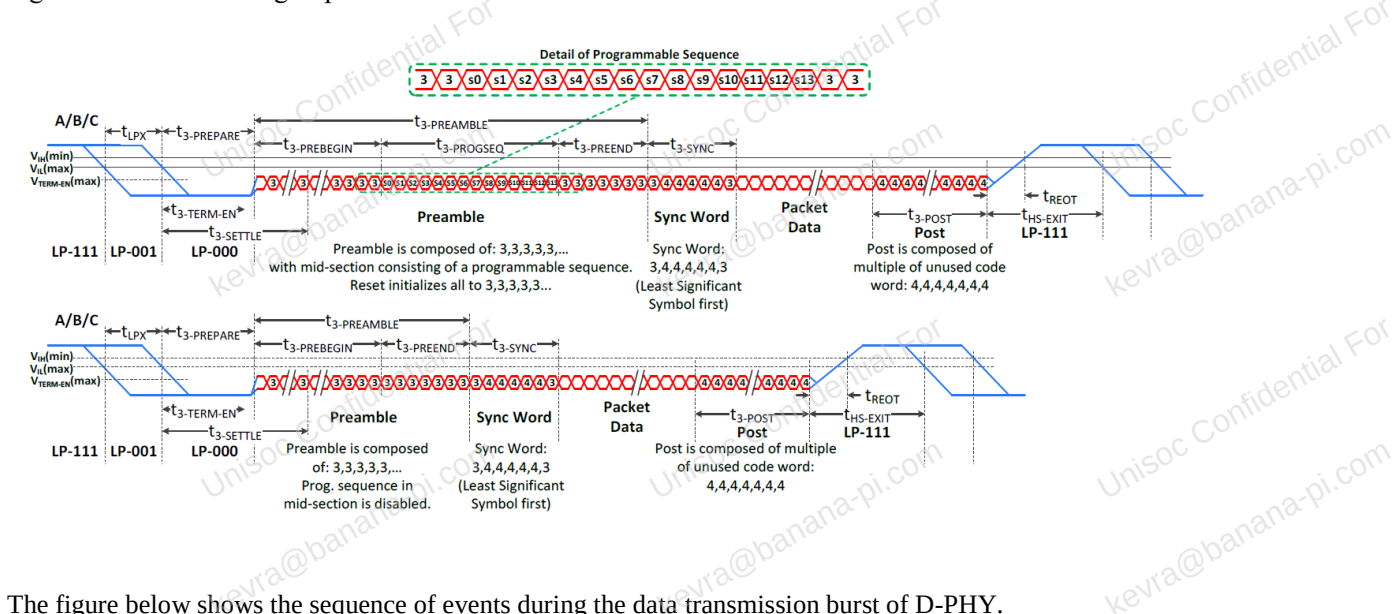
Table 10-3 Combo PHY PAD list

PAD Name	TYPE	Descriptions
AVDD_CSI_1P8	PIO	1.8V analog power supply
AVDD_CSI_1P2	PIO	1.2V analog power supply
AVSS_CSI	PIO	Analog ground
VSS_CSI	PIO	Common ground for ESD
PAD_CSI_A0_DP0_1P8	AIO	CPHY LANE0 A input pad, DPHY data lane0 P input
PAD_CSI_B0_DN0_1P8	AIO	CPHY LANE0 B input pad, DPHY data lane0 N input
PAD_CSI_C0_DP1_1P8	AIO	CPHY LANE0 C input pad, DPHY data lane1 P input

PAD Name	TYPE	Descriptions
PAD_CSI_A1_DN1_1P8	AIO	CPHY LANE1 A input pad, DPHY data lane1 N input
PAD_CSI_B1_CKP_1P8	AIO	CPHY LANE1 B input pad, DPHY clock lane P input
PAD_CSI_C1_CKN_1P8	AIO	CPHY LANE1 C input pad, DPHY clock lane N input
PAD_CSI_A2_DP2_1P8	AIO	CPHY LANE2 A input pad, DPHY data lane2 P input
PAD_CSI_B2_DN2_1P8	AIO	CPHY LANE2 B input pad, DPHY data lane2 N input
PAD_CSI_C2_DP3_1P8	AIO	CPHY LANE2 C input pad, DPHY data lane3 P input
PAD_CSI_NC_DN3_1P8	AIO	DPHY data lane3 N input

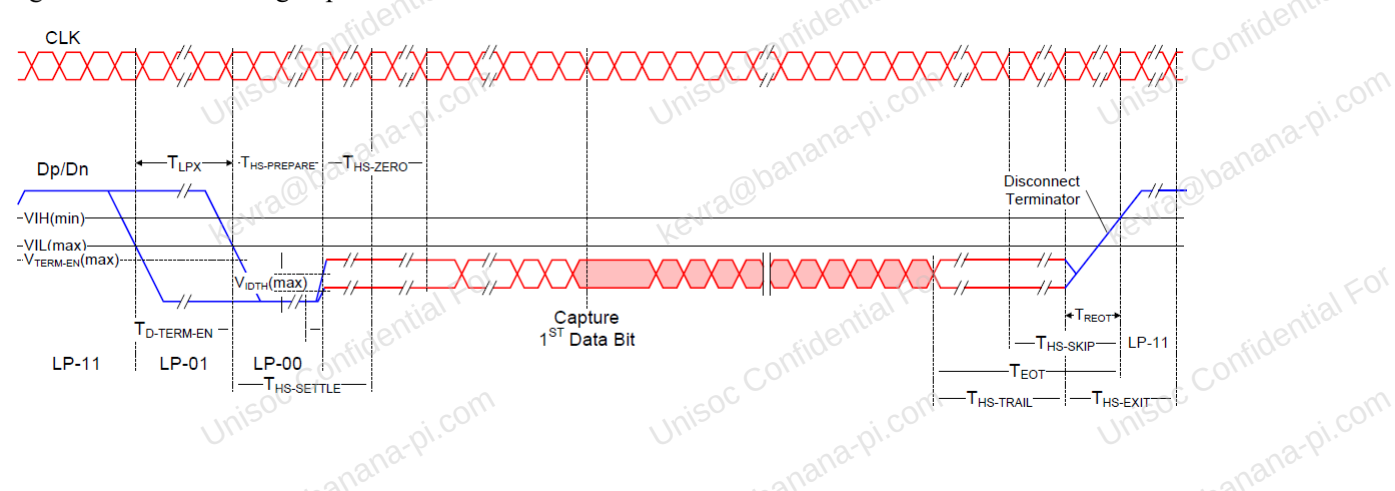
The figure below shows the sequence of events during the data transmission burst of C-PHY.

Figure 10-11 C-PHY High-Speed Data transmission in Burst



The figure below shows the sequence of events during the data transmission burst of D-PHY.

Figure 10-12 D-PHY High-Speed Data transmission in Burst



10.1.6.2 2p2lane D-PHY

The Combo PHY supports both MIPI C-PHY V1.2 and MIPI D-PHY 1.2 protocol, and there are three data

The 2p2lane D-PHY can be configured in three operation modes:

- CSI 4lane mode, one port
- CSI two ports mode, with two lanes of each port
- Debug log mode

This PHY support the following feature:

- TSMC7FFPlus process
- 1.8 V $\pm$ 10% Analog supply operation
- 0.63~0.935V Digital supply operation
- Operating junction temperature range: -40 °C ~ 125 °C
- Support one csi_4lane mode and two csi_2lane mode
- Data rate in high speed mode: ranging from 200 to 2500 Mbps per lane in all configured mode
- Data rate in low power mode: 10Mbps
- Protocol peripheral interface (PPI) for clock and data lanes

The PAD of the interface are listed in the table below:

Table 10-4 2p2lane D-PHY PAD list

Signal	I/O	Description
AVSS	PIO	Analog 1.8V Ground
AVDD18	PIO	Analog 1.8V power supply.
AVDD12	PIO	Analog 1.2V power supply.
VSS	PIO	Ground
DVDD11	PIO	Digital 0.75V power supply.
DVSS11	PIO	Digital Ground
CKN_M	AIO	Clock Lane N inout
CKP_M	AIO	Clock Lane P inout
DN0_M	AIO	Data lane0 N inout
DP0_M	AIO	Data lane0 P inout
DN1_M	AIO	Data lane1 N inout
DP1_M	AIO	Data lane1 P inout
CKN_S	AIO	Clock Lane N inout, of the second port
CKP_S	AIO	Clock Lane P inout, of the second port
DN0_S	AIO	Lane2 N inout, or data lane0 N inout of the second port
DP0_S	AIO	Lane2 P inout, or data lane0 P inout of the second port

Signal	I/O	Description
DN1_S	AIO	Lane3 N inout, or data lane1 N inout of the second port
DP1_S	AIO	Lane3 P inout, or data lane1 P inout of the second port

The timing of the HS data transmission is the same as shown in [Figure 10-12](#).

10.2 DPU_VSP

10.2.1 General Description

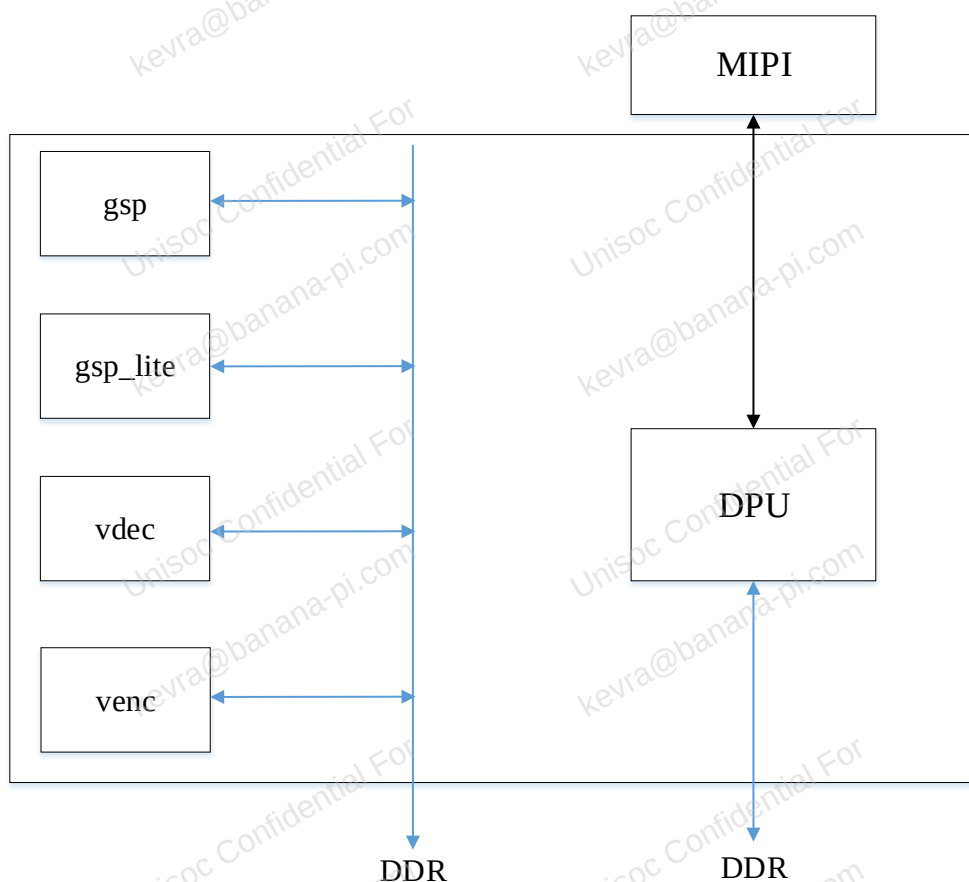
10.2.1.1 Overview

DPU_VSP system process display data, control and send display data to screen. DPU_VSP system completes raw video data compress and decompresses compressed video data.

10.2.1.2 Block Diagram

DPU_VSP mainly include DPU, dsi_ctrl, gsp, gsp_lite, venc and vdec.

Figure 10-13 DPU_VSP sys architecture



10.2.1.3 Features

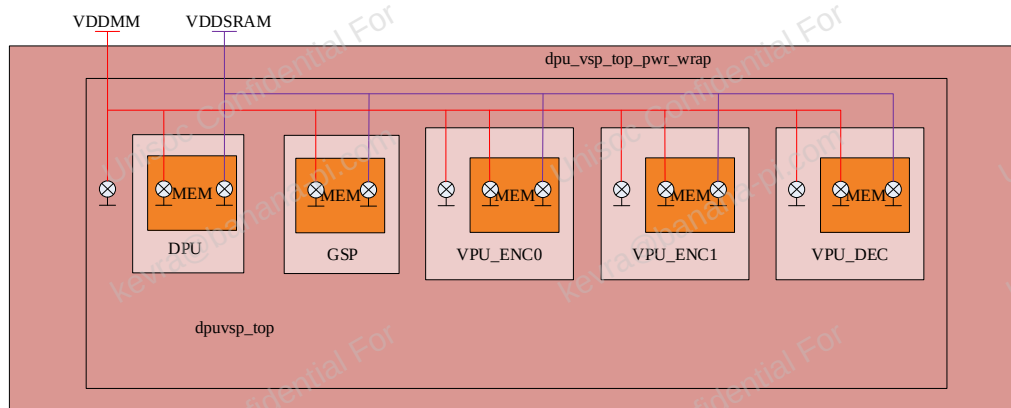
- DPU: dual core DPU, support two MIPI DSI interface, DSIv1.2 and DPHYv1.2, MIPI-DSI up to 4lane×2.5Gbps.
- DPU support resolution: QHD+@60fps/FHD+@120fps
- DPU support Frame Buffer Compression: AFBC1.2
- DPU support Display Stream Compression: DSC1.1.
- DPU support 8 layer alpha-blending and writeback.
- DPU support layer rotation and layer scaling.
- DPU support display calibrate:
 - Auto White Balance
 - Auto gamut calibrate
- DPU support ProtectView:
 - Blue Light Protection
 - Sun Light Protection
 - Low light protection
- DPU support ViveView:
 - Contrast enhancement
 - Sharpness enhancement
 - HDR playback
 - Wide Gamut display, sRGB/DCI-P3
- DPU support GreenView:
 - Ambient Adaptive Lightness
 - Content adaptive Brightness Control
- DPU support Display Tuning:
 - Hue/Saturation/lightness ration/lightness
 - 3D HSV/LCH
- DPU support DC/PWM brightness control for AMOLED.
- DPU support AOD(Always On Display) for AMOLED.
- DPU support Super Resolution.
- GSP has dual core GSP.
- GSP support color space conversion: support NV12,NV21,YV12,RGB interconvsn.
- GSP support scaling/rotation, 4 layer alpha-blending.
- H.264 Encoder: baseline profile, main profile, high profile, level 5.1, 4k@24fps 1080p@100fps, 720p@240fps
- H.264 Decoder: baseline profile, main profile, high profile, high 10 profile, level5.1, 4k@30fps
- H.265 Encoder: main profile, level5.1, 4k@60fps,1080p@240fps
- H.265 Decoder: main profile, main 10 profile, level5.1, 4k@30fps 10bit, 4k@60fps 8bit
- VP8 Encoder to 1080p@30fps by software
- VP8 Decoder up to 1080p@60fps
- VP9 Encoder to 1080p@30fps by software
- VP9 Decoder : profile0, profile2, 4k@30fps

- Support simultaneous instances of video encoder/decoder(scenario: miracast, video phone + miracast, video play[normal play window + small preview play window]).

10.2.1.4 System Low power Design

10.2.1.4.1 Power Scheme

Figure 10-14 DPU_VSP sys power architecture



Support clock configuration, for more detail please see register instruction.

Table 10-5 Power domain in DPU_VSP SYS

Power Domain Name	Modules
PD_DPU_VSP_SYS	The whole dpu_vsp sys except vpu_enc_top_pwr_wrap, vpu_dec_top_pwr_wrap, dpu0_top_pwr_wrap, dpu1_top_pwr_wrap
PD_VPU_ENC0	vpu_enc_top_pwr_wrap
PD_VPU_ENC1	vpu_enc_top_pwr_wrap
PD_VPU_DEC	vpu_dec_top_pwr_wrap

10.2.1.4.2 Power State

Table 10-6 Power states in DPU_VSP SYS

State	PD_DPU_VSP_SYS	PD_VPU_ENC0	PD_VPU_ENC1	PD_VPU_DEC
ALL_OFF	OFF	OFF	OFF	OFF
ALL_ON	ON	ON	ON	ON
VPUENC0_ON	ON	ON	OFF	OFF
VPUENC1_ON	ON	OFF	ON	OFF
VPUENC_ON	ON	ON	ON	OFF

State	PD_DPU_VSP_SYS	PD_VPU_ENC0	PD_VPU_ENC1	PD_VPU_DEC
VPUDECODE_ON	ON	OFF	OFF	ON

10.2.1.4.3 Low power mode

DPU_VSP system support low power mode:

- Matrix clock gate
Matrix clock gate is automatically gated when the bus is idle. This is controlled by LPC.
 - When lpc_eb is disable, matrix clock will not be gated.
 - When lpc_eb is enable, lpc is working.
- Light sleep
If all modules that can access ddr are stop in the DPU_VSP subsystem or busy signal is low, all buffer data to ddr has been transferred, and all lpc for ddr path are in low power state, the subsystem will be in light sleep mode.

10.2.2 DPU

10.2.2.1 Overview

The display processing unit (DPU) is HW composer and display output controller. The detail design information is presented below.

10.2.2.2 Features

- Update reg and lut update machine
- Support up to QHD+(3360 × 1440) Support up to 8 layers
- Support layer scaler down up to ½ x in both direction
- Support layer scaler up to 24x in both direction
- Support RGB/YUV/AFBC-RGB/AFBC-YUV data in each layer
- Support input rotation in each layer
- Support security accessing on top layer
- Support write back with/without AFBC
- Support up to 2 region checksum calculation
- Support dynamic ARQoS/AWQoS
- Support 24-bits DPI/eDPI output
- Support TE function when in eDPI output
- Support super resolution
- Support 2D HSV adjustment
- Support blue-light protector
- Support sun-light protector
- Support Ultra Dimming
- Support CABC
- Support dma req to move backlight data to PWM
- Support local tone mapping
- Support color adjustment

- Support gamma adjustment
- Support dithering
- Support 3D LUT
- Support round corner and fringe panel
- Support dpi V blanking timing adjustment
- 128bits AXI 3.0 master
- 32bits APB 3.0 slave

10.2.2.2.1 Data Format

The DPU supports YUV, RGB and AFBC source data, the data format used in module is listed in below, there are kinds of RGB/YUV format in source data, they MUST be adjusted to internal format through Byte switch and R/B switch.

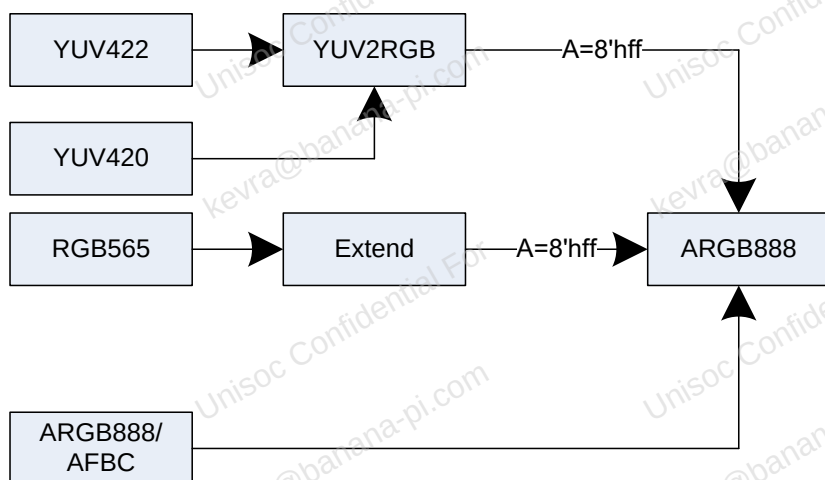
Table 10-7 Data Format List

Format	Code	Data Mapping (HW internal Format)																
ARGB888	4'b0000	<div><div>31241680</div><table><tr><td>A0</td><td>R0</td><td>G0</td><td>B0</td></tr><tr><td>A1</td><td>R1</td><td>G1</td><td>B1</td></tr></table></div>	A0	R0	G0	B0	A1	R1	G1	B1								
A0	R0	G0	B0															
A1	R1	G1	B1															
RGB565	4'b0001	<div><div>312821161250</div><table><tr><td>R1</td><td>G1</td><td>B1</td><td>R0</td><td>G0</td><td>B0</td></tr><tr><td>R3</td><td>G3</td><td>B3</td><td>R2</td><td>G2</td><td>B2</td></tr></table></div>	R1	G1	B1	R0	G0	B0	R3	G3	B3	R2	G2	B2				
R1	G1	B1	R0	G0	B0													
R3	G3	B3	R2	G2	B2													
YUV420 (2-plane)	4'b0010	<div><div>Y Plane</div><div>UV Plane</div><div><div>31241680</div><table><tr><td>Y3</td><td>Y2</td><td>Y1</td><td>Y0</td></tr><tr><td>Y7</td><td>Y6</td><td>Y5</td><td>Y4</td></tr></table><div><div>31241680</div><table><tr><td>V2</td><td>U2</td><td>V0</td><td>U0</td></tr><tr><td>V6</td><td>U6</td><td>V4</td><td>U4</td></tr></table></div></div></div>	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	V2	U2	V0	U0	V6	U6	V4	U4
Y3	Y2	Y1	Y0															
Y7	Y6	Y5	Y4															
V2	U2	V0	U0															
V6	U6	V4	U4															
YUV420 (3-plane)	4'b0011	<div><div>Y Plane</div><div>U Plane</div><div>V Plane</div><div><div>31241680</div><table><tr><td>Y3</td><td>Y2</td><td>Y1</td><td>Y0</td></tr><tr><td>Y7</td><td>Y6</td><td>Y5</td><td>Y4</td></tr></table><div><div>31241680</div><table><tr><td>U3</td><td>U2</td><td>U1</td><td>U0</td></tr></table><div><div>31241680</div><table><tr><td>V3</td><td>V2</td><td>V1</td><td>V0</td></tr></table></div></div></div></div>	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	U3	U2	U1	U0	V3	V2	V1	V0
Y3	Y2	Y1	Y0															
Y7	Y6	Y5	Y4															
U3	U2	U1	U0															
V3	V2	V1	V0															

Format	Code	Data Mapping (HW internal Format)
YUV422 (2-plane)	4'b0100	<div><div>Y Plane</div><div>UV Plane</div></div> <div><div><div>31241680</div><div>Y3Y2Y1Y0</div><div>Y7Y6Y5Y4</div></div><div><div>31241680</div><div>V2U2V0U0</div><div>V6U6V4U4</div></div></div>
AFBC- RGBA8888	4'b1000	<div><div>pxl_0~pxl_15</div><div><div>2345</div><div>1076</div><div>141589</div><div>13121110</div></div></div> <div>16 × 16 block</div>
AFBC-RGB565	4'b1001	<div><div>pxl_0~pxl_15</div><div><div>2345</div><div>1076</div><div>141589</div><div>13121110</div></div></div> <div>16 × 16 block</div>
AFBC-YUV420 2P	4'b1010	<div><div>Ypxl_0~pxl_32</div><div><div>345613141516</div><div>109811101918</div></div></div> <div><div>UVpxl_0~pxl_15</div><div><div>271217</div></div></div> <div>32 × 8 block</div>

The DPU supports many kinds of data format transform, in order to implement multi-layers blending, each layer data will be transformed to ARGB888 format, these transform are shown in below.

Figure 10-15 Intermediate Data Transform



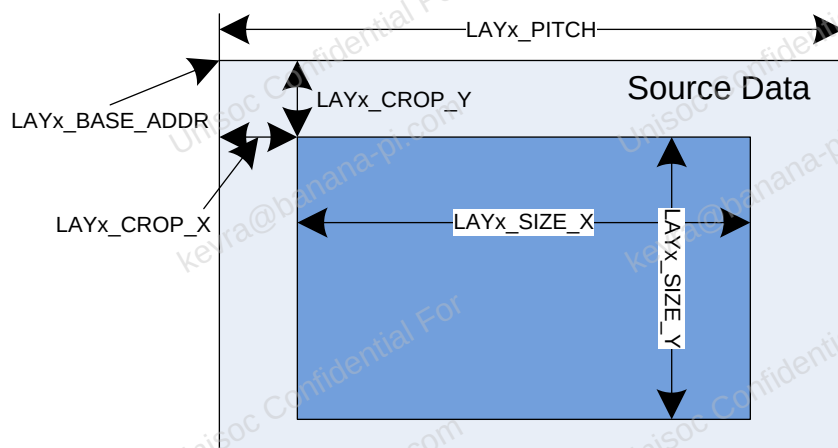
There are 2 extend modes to transform 5bits or 6bits data to 8bits,

- Padding with MSbs: $D[7:0] = \{D[4:0], D[4:2]\}$; $D[7:0] = \{D[5:0], D[5:4]\}$;
- Padding with 0s: $D[7:0] = \{D[4:0], 3'b0\}$; $D[7:0] = \{D[5:0], 2'b0\}$;

10.2.2.2.2 Data Fetch

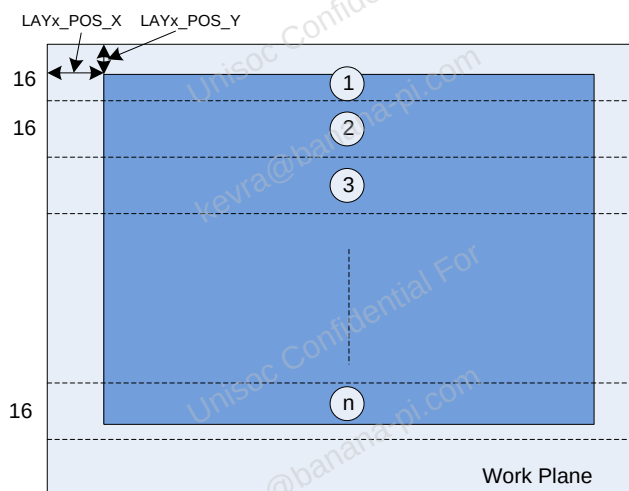
The DPU supports crop the layer data from source.

Figure 10-16 Data Crop



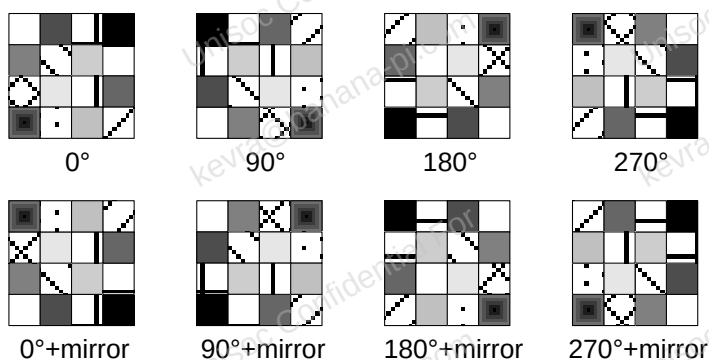
The DPU works with 16 lines slice, the layer will be split to slice data based on its display position, an example is shown in below.

Figure 10-17 Data Fetch Example



The DPU supports 8 input rotation.

Figure 10-18 Input Rotation



Rotation : left (anti-clockwise)
Mirror axis : horizontal line

10.2.2.2.3 YUV to RGB

The conversion supports ITU601, ITU709 and ITU2020 formula.

The constant coefficient for all formats is below.

Table 10-8 YUV2RGB Coefficient List

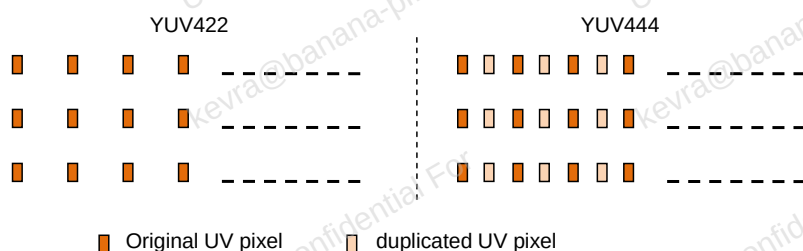
Standard	Full Range	Narrow Range
ITU601	$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4096 & 0 & 5743 & -7350 & 522048 \\ 4096 & -1410 & -2925 & 554840 & 2048 \\ 4096 & 7258 & 0 & -929038 & 2048 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \\ 1 \\ 1 \end{bmatrix} \gg 12$	$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4769 & 0 & 6537 & -913087 & 2048 \\ 4769 & -1605 & -3330 & 555316 & 2048 \\ 4769 & 8263 & 0 & -1133920 & 2048 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \\ 1 \\ 1 \end{bmatrix} \gg 12$

Standard	Full Range	Narrow Range
ITU709	$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4096 & 0 & 6450 & -8256492048 \\ 4096 & -767 & -1917 & 343644 & 2048 \\ 4096 & 7601 & 0 & -9728692048 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \\ 1 \\ 1 \end{bmatrix} \gg 1$	$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4769 & 0 & 7343 & -10162222048 \\ 4769 & -873 & -2183 & 314893 & 2048 \\ 4769 & 8652 & 0 & -11838162048 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \\ 1 \\ 1 \end{bmatrix} \gg 1$
ITU2020	$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4096 & 0 & 6040 & -7731152048 \\ 4096 & -674 & -2340 & 385827 & 2048 \\ 4096 & 7706 & 0 & -9863952048 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \\ 1 \\ 1 \end{bmatrix} \gg 1$	$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4769 & 0 & 6876 & -9564182048 \\ 4769 & -767 & -2664 & 362913 & 2048 \\ 4769 & 8773 & 0 & -11992152048 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \\ 1 \\ 1 \end{bmatrix} \gg 1$

As above formula, the input data is YUV444, so the fetched data must be up-sampled to YUV444 before converse.

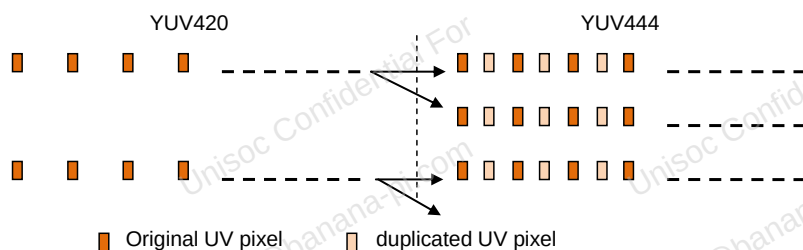
When the input is YUV422, you need up-sample UV in horizontal.

Figure 10-19 YUV422 to YUV444 Conversion



When the input is YUV420, you need up-sample UV in horizontal, and duplicate UV in vertical.

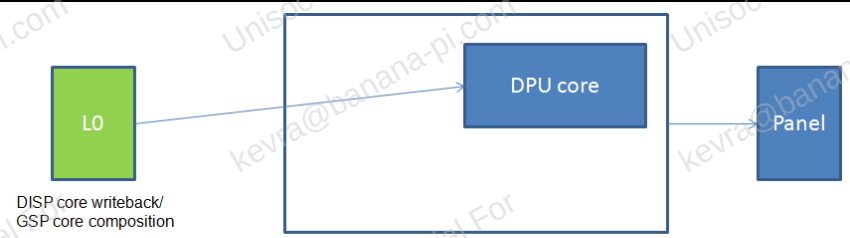
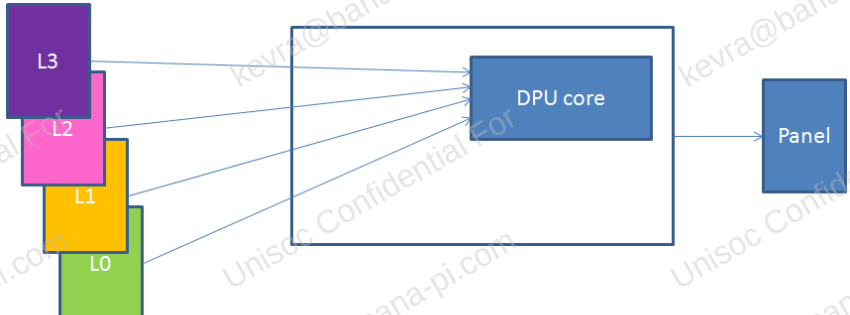
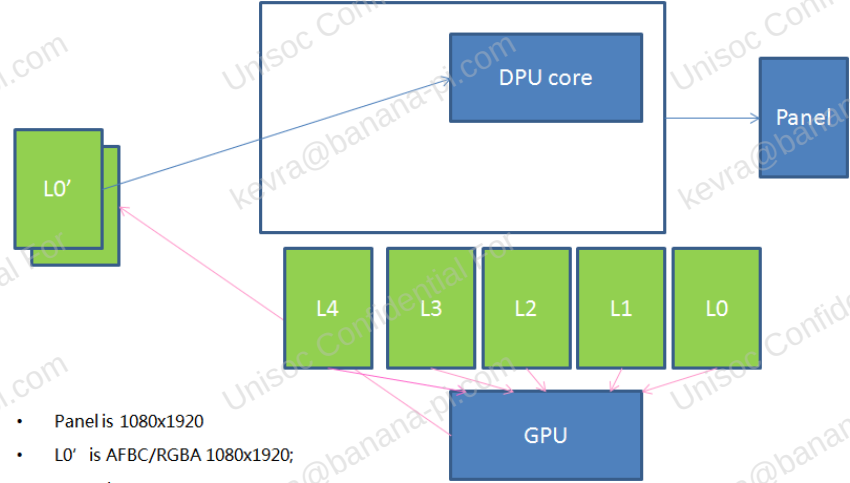
Figure 10-20 YUV420 to YUV444 Conversion

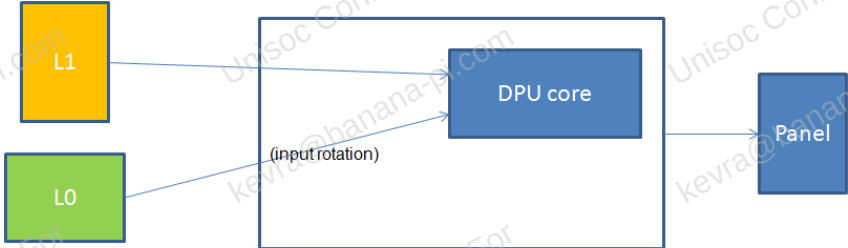
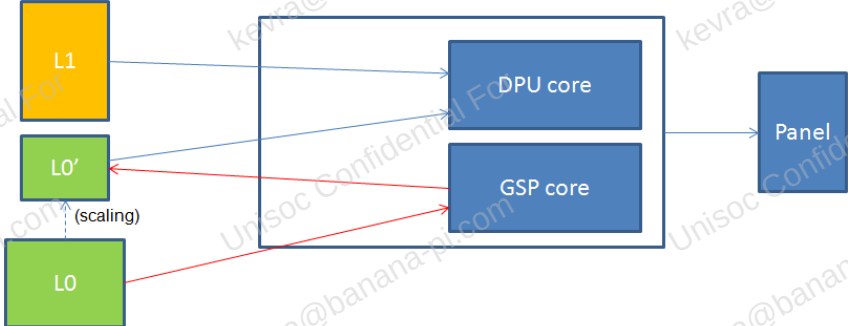
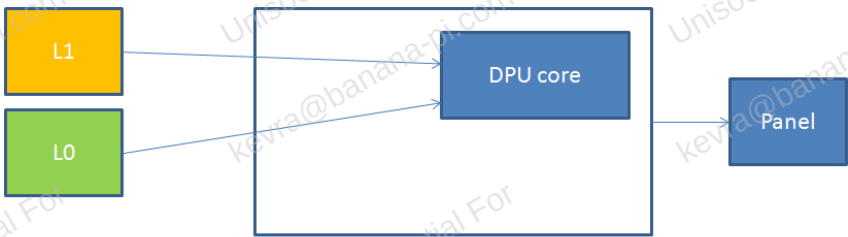


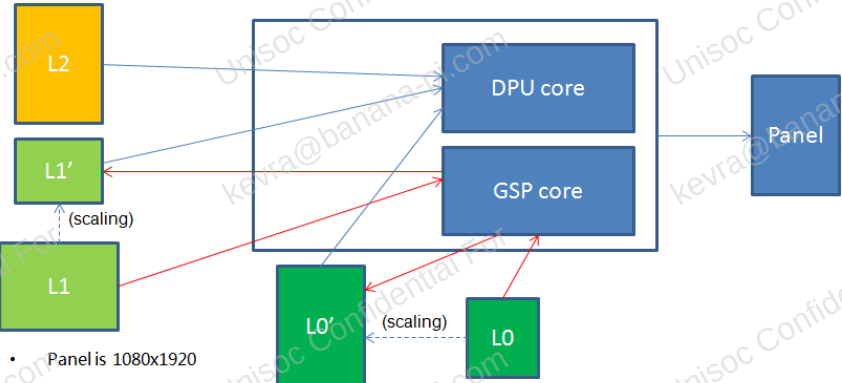
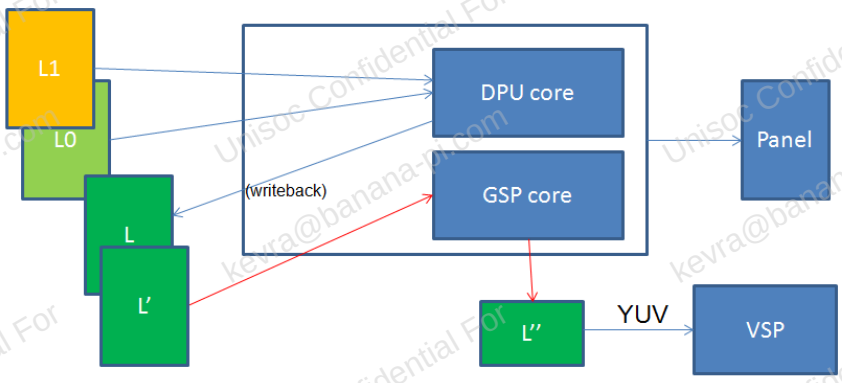
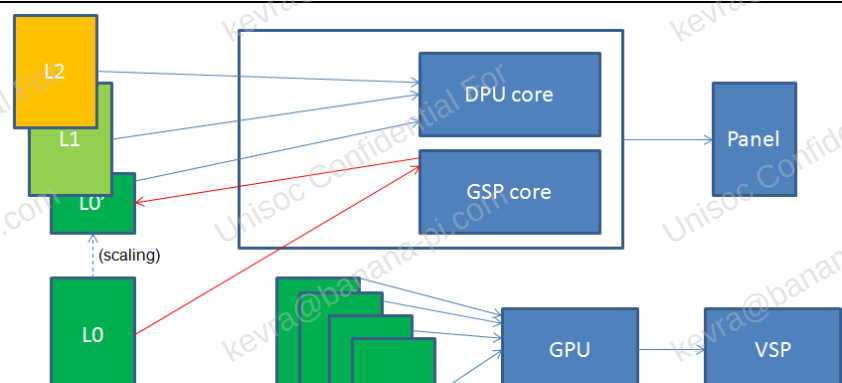
10.2.2.2.4 Scenario Path

The DPU includes many application, the typical ones are shown in below table.

Table 10-9 Typical Data Path

Scenario	Data Path
LCD ON	 <p>DISP core writeback/ GSP core composition</p> <ul style="list-style-type: none"> Panel is 1080x1920 L0 is AFBC/RGBA 1080x1920;
UI (<=4 full layer)	 <ul style="list-style-type: none"> Panel is 1080x1920 L0 is AFBC/RGBA 1080x1920; L1 is AFBC/RGBA 1080x1920; L2 is AFBC/RGBA 1080x1920; L3 is AFBC/RGBA 1080x1920;
UI (>4 full layer)	 <ul style="list-style-type: none"> Panel is 1080x1920 L0' is AFBC/RGBA 1080x1920; L0~L4 is AFBC/RGBA 1080x1920;

Scenario	Data Path
Camera Preview	 <ul style="list-style-type: none"> Panel is 1080x1920 L0 is YUV420 1920x1080; L1 is AFBC/RGBA 1080x1920; DPU core needs to co-work with GSP core if L0 isn't FHD;
Video Playback (FHD @ 30fps, Portrait Mode)	 <ul style="list-style-type: none"> Panel is 1080x1920 L0 is YUV420 1920x1080; L1 is AFBC/RGBA 1080x1920; DPU core co-works with GSP core; GSP core runs one time every 2 frame;
Video Playback (FHD @ 30fps, Landscape Mode)	 <ul style="list-style-type: none"> Panel is 1920x1080 L0 is YUV420 1920x1080; L1 is AFBC/RGBA 1920x1080;

Scenario	Data Path
VoLTE	 <p>The diagram shows the data path for VoLTE. It includes a DPU core and a GSP core. L2 (AFBC/RGBA 1080x1920) feeds into the DPU core. L1 (YUV420 480x640) feeds into the GSP core. L1' (YUV420 1080x1920) is a scaled version of L1. L0 (YUV420 1080x1920) feeds into the GSP core. L0' (YUV420 1080x1920) is a scaled version of L0. The DPU core and GSP core co-work, with the GSP core running one time every 2 frames. The output goes to the Panel.</p> <ul style="list-style-type: none"> Panel is 1080x1920 L0 is YUV420 1080x1920; L1 is YUV420 480x640; L2 is AFBC/RGBA 1080x1920; DPU core co-works with GSP core; GSP core runs one time every 2 frame;
Wi-Fi Display (Clone Mode)	 <p>The diagram shows the data path for Wi-Fi Display in Clone Mode. It includes a DPU core and a GSP core. L1 (AFBC/RGBA 1080x1920) feeds into the DPU core. L0 (AFBC/RGBA 1080x1920) feeds into the DPU core. L' (AFBC/RGBA 1080x1920) feeds into the GSP core. L'' (YUV 1280x720) is the output of the GSP core, which is then processed by the VSP. The DPU core and GSP core co-work when input needs scaling. The output goes to the Panel.</p> <ul style="list-style-type: none"> Local Panel is 1080x1920 L0 is AFBC/RGBA 1080x1920; L1 is AFBC/RGBA 1080x1920; DPU core needs to co-work with GSP core when input needs scaling; Remote Panel is 1280x720; Input L' is AFBC 1080x1920; Output L'' is YUV 1280x720; GSP core → Remote; L' is one frame delay;
Wi-Fi Display (Extend Mode)	 <p>The diagram shows the data path for Wi-Fi Display in Extend Mode. It includes a DPU core, a GSP core, and a GPU. L2 (AFBC/RGBA 1080x1920) feeds into the DPU core. L1 (AFBC/RGBA 1080x1920) feeds into the DPU core. L0' (AFBC/RGBA 1080x1920) is a scaled version of L1. L0 (AFBC/RGBA 1080x1920) feeds into the GPU. L' (AFBC/RGBA 1080x1920) feeds into the GPU. The DPU core and GSP core co-work, with the GSP core running one time every 2 frames. The output goes to the Panel.</p> <ul style="list-style-type: none"> DPU core + GSP core → Local; GPU → Remote

10.2.2.3 Control Registers

See *UIS7885 DPU VSP System Control Registers Instructions*.

10.2.3 DSI controller

10.2.3.1 Overview

The Display Serial Interface (DSI) is part of a group of communication protocols defined by the MIPI Alliance. The DSI_CTRL_TOP is a digital core that implements protocol level function and lane management functions defined in the MIPI DSI Specification. The DSI_CTRL_TOP provides an interface between the display ctrl module and the MIPI D-PHY.

10.2.3.2 Features

The DSI_CTRL_TOP has the following main features:

- Support DPI interface with Display ctrl module
- Support APB 2.0 interface to configure registers.
- Support video mode transmit and command mode transmit
- Support all the video format in DSI v1.3 Spec
- Support all the packet types in DSI v1.3 Spec
- Support De-skew function in DSI v1.3 Spec
- Support all DCS commands in DCS v1.2 Spec
- Support command transmit in Vertical blanking region of video mode
- Support DPHY clock lane SSC(Spread Spectrum Clocking)

10.2.3.3 Standards Compliance

The mipi_dsi_host conforms to the following standards:

- MIPI® Alliance Specification for Display Serial Interface (DSI) v1.2 - 18 June 2014
- MIPI® Alliance Specification for Display Command Set (DCS) v1.2 - 18 June 2014
- MIPI® Alliance Standard for Display Pixel Interface v2.00 (DPI-2) - 15 September 2005
- MIPI® Alliance Specification for Stereoscopic Display Formats (SDF) v1.0 - 22 November 2011
- MIPI® Alliance Specification for D-PHY v1.2 - 14 September 2014
- AMBA® 2 Specification (APB) from Arm® processor

10.2.3.4 Control Registers

See *UIS7885 DPU VSP System Control Registers Instructions*.

10.2.4 GSP

10.2.4.1 Overview

GSP is hardware module for image rotation, blending and scaling. It includes 2 cores.

10.2.4.2 Features

- Support ARQOS/AWQOS adopt adjustment in AXI bus.
- Support AXI clear when axi bus hanging issue.

- Support 1core1task/2core1task.
 - 1core1task: single core support 4 layer compositions (include layer0/layer1 scaler&rotation, and input yuv).
 - 2core1task: two core support only layer0 scaler (include rotation, input yuv).
- Support AFBCD/AFBCE 1.2 function:
 - Support little endian in AFBCE/AFBCD.
 - Support XRGB888(blk_size = 16 × 16, 8bit)/RGB565(blk_size = 16 × 16, 8bit)/YUV420_2P(blk_size = 32 × 8, 8bit)/ YUV420_2P(blk_size=32 × 8, 10bit HDR plus10 mode) AFBCD.
 - Support ARGB888(blk_size=16 × 16, 8bit)/YUV420_2P(blk_size=32 × 8, 8bit) AFBCE.
 - Support AWLEN align mode(64Byte/32Byte/16Byte) for header/data part in AFBC output mode, depended on single channel(lpddr4) or two channel(lpddr4/lpddr3) used.
 - Need 16 pixel align in width and height of ARGB888/RGB565 in AFBCD mode.
 - Need 32 pixel align in width and 8 pixel align in height of YUV420-2p in AFBCD mode.
 - Need 16 × 16 aligned in workarea size/start(in the case that “bk_en=0, destination size is workarea”) and des_pitch(height) when ARGB888 in AFBCE mode.
 - Need 32x8 aligned in workarea size/start(in the case that “bk_en=0, destination size is workarea”) and des_pitch(height) when YUV420_2P in AFBCE mode.
 - Support ARGB888/RGB565/YUV420_2P AFBCD input rotation.
 - Support ARGB888 AFBCE output rotation.
 - Not support YUV420_2P AFBCE output rotation.
- Support input rotation function:
 - Support input rotation : 0,90,180 and 270 degree rotation, flip horizontal and flip vertical.
 - Support Layer0/1 YUV420-2p/YUV420-3p/YV12/RGB888/ARGB888/RGB565 input rotation whether scaling or not.
 - Not support Layer2/3 input rotation.
 - Not support Layer0/1 input rotation in pallet mode or yuv422 format.
- Support blending function:
 - Support 5 Layer alpha z-order blending and color key, including Layer0/1/2/3 and background layer.
 - Support background layer fill.
- Support scaling function:
 - Support ARGB input scaling.
 - Support scaling vertical/horizontal tap : 4/6/8.
 - Support Layer0/1 scaling from 1/16~24 in normal mode.
 - Support Layer0/1 scaling from 1/4~24 in input rotation mode.
 - Support Layer0/1 scaling from 1/4~24 in AFBCD input mode.
 - Need 2pixel aligned in x/y direction of clip_start/clip_size/pitchx/heighx when scaling 1/2 down sample.
 - Need 4pixel aligned in x/y direction of clip_start/clip_size/pitchx/heighx when scaling 1/4 down sample.
 - Support scaling coefficient load from memory from APB register.
- Support input layer feature:
 - Support input size: 8191x8191(clip picture).
 - Support clipping.
 - Support 4 input layer.

- Support Layer0~3 register shadow.
- Support configures Layer3 by APB in secure mode.
- Support image data format in Layer0/1:
 - 3-plane: YUV420, YV12 (YUV420_3p UV 16align).
 - 2-plane: YUV422, YUV420, YUV420 (HDR plus10 mode).
 - 1-plane: ARGB888, RGB888, RGB565.
- Support image data format in Layer2/3:
 - 1-plane: ARGB888, RGB888, RGB565.
- Support contrast, brightness and saturation adjustment in Layer0/1 YUV mode.
- Support 6 color space convert matrix for yuv to rgb in Layer0/1 YUV mode.
- Force dither in HDR plus10 mode, to change data from 10bit to 8bit.
- Force RGB output from HDR submodule in HDR plus10 mode.
- Need even pitch in Layer0/1 YUV mode.
- Support RGB/alpha swap mode except layerx in pallet mode.
- Support input data endian configures.
- Support output Layer feature:
 - Support output size: 8191 × 8191.
 - Support destination register shadow.
 - Support output rotation: 0,90,180 and 270 degree rotation, flip horizontal and flip vertical.
 - Support 6 color space convert matrix for rgb to yuv in destination output.
 - Support dithering in RGB565 mode.
 - Support dithering in RGB666 mode.
 - Support output endian configures.
 - Support image data format:
 - 3-plane: YUV420.
 - 2-plane: YUV422, YUV420.
 - 1-plane: ARGB888, RGB888 (32bit), RGB565, RGB666.
- Support register configured by software configure mode.

Table 10-10 source/destination data formats

Format	Data Mapping(big endian)																												
YUV422_2p	<div>1-plane:</div> <table><tr><th>[31:24]</th><th>[23:16]</th><th>[15:8]</th><th>[7:0]</th></tr><tr><td>Y3</td><td>Y2</td><td>Y1</td><td>Y0</td></tr><tr><td>Y7</td><td>Y6</td><td>Y5</td><td>Y4</td></tr></table> <div>2-plane:</div> <table><tr><th>[31:24]</th><th>[23:16]</th><th>[15:8]</th><th>[7:0]</th></tr><tr><td>V1</td><td>U1</td><td>V0</td><td>U0</td></tr><tr><td>V3</td><td>U3</td><td>V2</td><td>U2</td></tr></table> <div>The U/V data is half of the Y data in horizontal. Data address should be half-word aligned,(include 2 pixels)</div>	[31:24]	[23:16]	[15:8]	[7:0]	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	[31:24]	[23:16]	[15:8]	[7:0]	V1	U1	V0	U0	V3	U3	V2	U2				
[31:24]	[23:16]	[15:8]	[7:0]																										
Y3	Y2	Y1	Y0																										
Y7	Y6	Y5	Y4																										
[31:24]	[23:16]	[15:8]	[7:0]																										
V1	U1	V0	U0																										
V3	U3	V2	U2																										
YUV420_2p	<div>1-plane:</div> <table><tr><th>[31:24]</th><th>[23:16]</th><th>[15:8]</th><th>[7:0]</th></tr><tr><td>Y3</td><td>Y2</td><td>Y1</td><td>Y0</td></tr><tr><td>Y7</td><td>Y6</td><td>Y5</td><td>Y4</td></tr></table> <div>2-plane:</div> <table><tr><th>[31:24]</th><th>[23:16]</th><th>[15:8]</th><th>[7:0]</th></tr><tr><td>V1</td><td>U1</td><td>V0</td><td>U0</td></tr><tr><td>V3</td><td>U3</td><td>V2</td><td>U2</td></tr></table> <div>The U/V data is half of the Y data in horizontal and vertical. Data address should be half-word aligned,(include 2 pixels)</div>	[31:24]	[23:16]	[15:8]	[7:0]	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	[31:24]	[23:16]	[15:8]	[7:0]	V1	U1	V0	U0	V3	U3	V2	U2				
[31:24]	[23:16]	[15:8]	[7:0]																										
Y3	Y2	Y1	Y0																										
Y7	Y6	Y5	Y4																										
[31:24]	[23:16]	[15:8]	[7:0]																										
V1	U1	V0	U0																										
V3	U3	V2	U2																										
YUV420_2p (HDR plus10mode)	<div>1-plane:</div> <table><tr><th>[31:26]</th><th>[25:16]</th><th>[15:10]</th><th>[9:0]</th></tr><tr><td>Reserved</td><td>Y1</td><td>Reserved</td><td>Y0</td></tr><tr><td>Reserved</td><td>Y3</td><td>Reserved</td><td>Y2</td></tr></table> <div>2-plane:</div> <table><tr><th>[31:26]</th><th>[25:16]</th><th>[15:10]</th><th>[9:0]</th></tr><tr><td>Reserved</td><td>V0</td><td>Reserved</td><td>U0</td></tr><tr><td>Reserved</td><td>V1</td><td>Reserved</td><td>U1</td></tr><tr><td>Reserved</td><td>V2</td><td>Reserved</td><td>U2</td></tr></table>	[31:26]	[25:16]	[15:10]	[9:0]	Reserved	Y1	Reserved	Y0	Reserved	Y3	Reserved	Y2	[31:26]	[25:16]	[15:10]	[9:0]	Reserved	V0	Reserved	U0	Reserved	V1	Reserved	U1	Reserved	V2	Reserved	U2
[31:26]	[25:16]	[15:10]	[9:0]																										
Reserved	Y1	Reserved	Y0																										
Reserved	Y3	Reserved	Y2																										
[31:26]	[25:16]	[15:10]	[9:0]																										
Reserved	V0	Reserved	U0																										
Reserved	V1	Reserved	U1																										
Reserved	V2	Reserved	U2																										

Format	Data Mapping(big endian)											
YUV420_3p	1-plane:											
	[31:24]		[23:16]		[15:8]		[7:0]					
	Y3		Y2		Y1		Y0					
	Y7		Y6		Y5		Y4					
	2-plane:											
	[31:24]		[23:16]		[15:8]		[7:0]					
	U3		U2		U1		U0					
	U7		U6		U5		U4					
	3-plane:											
	[31:24]		[23:16]		[15:8]		[7:0]					
	V3		V2		V1		V0					
	V7		V6		V5		V4					
	The U/V data is half of the Y data in horizontal and vertical.											
	Data address should be half-word aligned,(include 2 pixels)											
RGB888	[31:24]		[23:16]		[15:8]		[7:0]					
	Reserved		R0		G0		B0					
	Reserved		R1		G1		B1					
ARGB888	[31:24]		[23:16]		[15:8]		[7:0]					
	A0		R0		G0		B0					
	A1		R1		G1		B1					
RGB666	[31:24]		[23:18]		[15:10]		[7:2]					
	Reserved		R0		G0		B0					
	Reserved		R1		G1		B1					
RGB565	[31:27]		[26:21]		[20:16]		[15:11]		[10:5]		[4:0]	
	R1		G1		B1		R0		G0		B0	
	R3		G3		B3		R2		G2		B2	

Format	Data Mapping(big endian)				
	All input/output data support 4 endian mode(32bit)				
	endian mode	[31:24] Byte3	[23:16] Byte2	[15:8] Byte1	[7:0] Byte0
	00	Byte3	Byte2	Byte1	Byte0
	01	Byte0	Byte1	Byte2	Byte3
	10	Byte2	Byte3	Byte0	Byte1
	11	Byte1	Byte0	Byte3	Byte2
	All input/output data support 1 endian mode(64bit)				
	endian mode	[63:32] Byte7-4	[31:0] Byte3-0		
	0	Byte7-4	Byte3-0		
	1	Byte3-0	Byte7-4		
	RGB configure				
	RGB mode	Sequence			
	000	RGB			
	001	RBG			
	010	GRB			
	011	GBR			
	100	BGR			
	101	BRG			
	ARGB 888 configure				
	ARGB888 mode	Sequence			
	0	ARGB			
	1	RGBA			

10.2.5 GSP_lite

10.2.5.1 Overview

GSP is hardware module for image rotation, blending and scaling. It is single core GSP.

10.2.5.2 Features

- Support ARQOS/AWQOS adopt adjustment in AXI bus.
- Support only 1core 1task mode.

1core1task: single core support 4 layer compositions (include layer0/layer1 scaler&rotation, and input yuv).

- Support AFBCE/AFBCD 1.2 function:
 - Support little endian in AFBCE/AFBCD.
 - Support ARGB888(blk_size = 16×16), XRGB888(blk_size = 16×16), RGB565(blk_size = 16×16), YUV420_2P (blk_size = 32×8) AFBCD.
 - Support ARGB888(blk_size = 16×16)/YUV420_2P(blk_size = 32×8) AFBCE.
 - Support AWLEN align mode(64Byte/32Byte/16Byte) for header/data part in AFBC output mode, depended on single channel(lpddr4) or two channel(lpddr4/lpddr3) used.
 - Need 16pixel align in width and height of ARGB888/RGB565 in AFBCD mode.
 - Need 32pixel align in width and 8 pixel align in height of YUV420-2p in AFBCD mode.
 - Need 16×16 aligned in workarea size/start(in the case that "bk_en=0, destination size is workarea") and des_pitch(height) when ARGB888 in AFBCE mode.
 - Need 32×8 aligned in workarea size/start(in the case that "bk_en=0, destination size is workarea") and des_pitch(height) when YUV420_2P in AFBCE mode.
 - Support ARGB888/RGB565/YUV420_2P AFBCD input rotation.
 - Support ARGB888 AFBCE output rotation.
 - Not support YUV420_2P AFBCE output rotation.
 - Not support 10bit AFBCE/AFBCD.
- Support input rotation function:
 - Support input rotation : 0,90,180 and 270 degree rotation, flip horizontal and flip vertical.
 - Support Layer0/1 YUV420-2p/YUV420-3p/YV12 input rotation whether scaling or not.
 - Support Layer0 RGB888/ARGB888/RGB565 input rotation in no scaling.
 - Support Layer1 RGB888/ARGB888/RGB565 input rotation whether scaling or not.
 - Not support Layer2/Layer3 OSD layers input rotation.
 - Not support rotation if layer0/1 in pallet mode or yuv422 format.
- Support blending function:
 - Support 5 Layer alpha z-order blending and color key, including Layer0/1/2/3 and background layer.
 - Support background layer fill.
- Support scaling function:
 - Support ARGB input scaling.
 - Support scaling vertical/horizontal tap : 4/6/8.
 - Support Layer0/1 scaling from 1/16~24 in normal mode.
 - Support Layer0/1 scaling from 1/4~24 in input rotation mode.
 - Support Layer0/1 scaling from 1/4~24 in AFBCD input mode.
 - Need 2pixel aligned in x/y direction of clip_start/clip_size/pitchx/heighx when scaling 1/2 down sample.
 - Need 4pixel aligned in x/y direction of clip_start/clip_size/pitchx/heighx when scaling 1/4 down sample.
 - Support scaling coefficient load from memory from APB register.
- Support input layer feature:
 - Support input size: 8191×8191 (clip picture).
 - Support clipping.
 - Support 4 input layer.

- Support Layer0~3 register shadow.
- Support configures Layer3 by APB in secure mode.
- Support image data format in Layer0/1:
 - 3-plane: YUV420, YV12 (YUV420_3p UV 16align).
 - 2-plane: YUV422, YUV420.
 - 1-plane: ARGB888, RGB888, RGB565.
- Support image data format in Layer2/3:
 - 1-plane: ARGB888, RGB888, RGB565.
- Support contrast, brightness and saturation adjustment in Layer0/1 YUV mode.
- Support 6 color space convert matrix for yuv to rgb in Layer0/1 YUV mode.
- Need even pitch in Layer0/1 YUV mode.
- Support RGB/alpha swap mode except layerx in pallet mode.
- Support input data endian configures.
- Support output Layer feature:
 - Support output size: 8191 × 8191.
 - Support destination0/1 register shadow.
 - Support output rotation: 0,90,180 and 270 degree rotation, flip horizontal and flip vertical.
 - Support 6 color space convert matrix for rgb to yuv in destination output.
 - Support RGB565 dithering in RGB565 mode.
 - Support RGB666 dithering in RGB666 mode.
 - Support output endian configures.
 - Support image data format:
 - 3-plane: YUV420.
 - 2-plane: YUV422, YUV420.
 - 1-plane: ARGB888, RGB888 (32bit), RGB565, RGB666.
- Support register configured by software configure mode.

Table 10-11 source/destination data format

Format	Data Mapping(big endian)												
YUV422	1-plane:												
	<table><tr><th>[31:24]</th><th>[23:16]</th><th>[15:8]</th><th>[7:0]</th></tr><tr><td>Y3</td><td>Y2</td><td>Y1</td><td>Y0</td></tr><tr><td>Y7</td><td>Y6</td><td>Y5</td><td>Y4</td></tr></table>	[31:24]	[23:16]	[15:8]	[7:0]	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4
	[31:24]	[23:16]	[15:8]	[7:0]									
	Y3	Y2	Y1	Y0									
	Y7	Y6	Y5	Y4									
	2-plane:												
	<table><tr><th>[31:24]</th><th>[23:16]</th><th>[15:8]</th><th>[7:0]</th></tr><tr><td>V1</td><td>U1</td><td>V0</td><td>U0</td></tr><tr><td>V3</td><td>U3</td><td>V2</td><td>U2</td></tr></table>	[31:24]	[23:16]	[15:8]	[7:0]	V1	U1	V0	U0	V3	U3	V2	U2
	[31:24]	[23:16]	[15:8]	[7:0]									
	V1	U1	V0	U0									
	V3	U3	V2	U2									
The U/V data is half of the Y data in horizontal.													
Data address should be half-word aligned,(include 2 pixels)													

Format	Data Mapping(big endian)																																								
YUV420_2p	<table><tr><td rowspan="6">Y</td><td>[7:0]</td><td rowspan="3">UV</td><td>[15:8]</td><td>[7:0]</td></tr><tr><td>Y0</td><td>V0</td><td>U0</td></tr><tr><td>Y1</td><td>V2</td><td>U2</td></tr><tr><td>Y2</td><td></td><td>V4</td><td>U4</td></tr><tr><td>....</td><td></td><td></td><td></td></tr><tr><td>Y7</td><td></td><td>V14</td><td>U14</td></tr></table> <p>The U/V data is half of the Y data in horizontal and vertical. Data address should be half-word aligned,(include 2 pixels)</p>					Y	[7:0]	UV	[15:8]	[7:0]	Y0	V0	U0	Y1	V2	U2	Y2		V4	U4					Y7		V14	U14													
Y	[7:0]	UV	[15:8]	[7:0]																																					
	Y0		V0	U0																																					
	Y1		V2	U2																																					
	Y2		V4	U4																																					
																																									
	Y7		V14	U14																																					
YUV420_3p	<p>1-plane:</p> <table><tr><td>[31:24]</td><td>[23:16]</td><td>[15:8]</td><td>[7:0]</td></tr><tr><td>Y3</td><td>Y2</td><td>Y1</td><td>Y0</td></tr><tr><td>Y7</td><td>Y6</td><td>Y5</td><td>Y4</td></tr></table> <p>2-plane:</p> <table><tr><td>[31:24]</td><td>[23:16]</td><td>[15:8]</td><td>[7:0]</td></tr><tr><td>U3</td><td>U2</td><td>U1</td><td>U0</td></tr><tr><td>U7</td><td>U6</td><td>U5</td><td>U4</td></tr></table> <p>3-plane:</p> <table><tr><td>[31:24]</td><td>[23:16]</td><td>[15:8]</td><td>[7:0]</td></tr><tr><td>V3</td><td>V2</td><td>V1</td><td>V0</td></tr><tr><td>V7</td><td>V6</td><td>V5</td><td>V4</td></tr></table> <p>The U/V data is half of the Y data in horizontal and vertical. Data address should be half-word aligned,(include 2 pixels)</p>					[31:24]	[23:16]	[15:8]	[7:0]	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	[31:24]	[23:16]	[15:8]	[7:0]	U3	U2	U1	U0	U7	U6	U5	U4	[31:24]	[23:16]	[15:8]	[7:0]	V3	V2	V1	V0	V7	V6	V5	V4
[31:24]	[23:16]	[15:8]	[7:0]																																						
Y3	Y2	Y1	Y0																																						
Y7	Y6	Y5	Y4																																						
[31:24]	[23:16]	[15:8]	[7:0]																																						
U3	U2	U1	U0																																						
U7	U6	U5	U4																																						
[31:24]	[23:16]	[15:8]	[7:0]																																						
V3	V2	V1	V0																																						
V7	V6	V5	V4																																						
RGB888	<table><tr><td>[31:24]</td><td>[23:16]</td><td>[15:8]</td><td>[7:0]</td><td></td></tr><tr><td>Reserved</td><td>R0</td><td>G0</td><td>B0</td><td></td></tr><tr><td>Reserved</td><td>R1</td><td>G1</td><td>B1</td><td></td></tr></table>					[31:24]	[23:16]	[15:8]	[7:0]		Reserved	R0	G0	B0		Reserved	R1	G1	B1																						
[31:24]	[23:16]	[15:8]	[7:0]																																						
Reserved	R0	G0	B0																																						
Reserved	R1	G1	B1																																						
ARGB888	<table><tr><td>[31:24]</td><td>[23:16]</td><td>[15:8]</td><td>[7:0]</td><td></td></tr><tr><td>A0</td><td>R0</td><td>G0</td><td>B0</td><td></td></tr><tr><td>A1</td><td>R1</td><td>G1</td><td>B1</td><td></td></tr></table>					[31:24]	[23:16]	[15:8]	[7:0]		A0	R0	G0	B0		A1	R1	G1	B1																						
[31:24]	[23:16]	[15:8]	[7:0]																																						
A0	R0	G0	B0																																						
A1	R1	G1	B1																																						

Format	Data Mapping(big endian)																																																											
RGB666	[31:24]	[23:18]	[15:10]	[7:2]																																																								
	Reserved	R0	G0	B0																																																								
	Reserved	R1	G1	B1																																																								
RGB565	[31:27]	[26:21]	[20:16]	[15:11]	[10:5]	[4:0]																																																						
	R1	G1	B1	R0	G0	B0																																																						
	R3	G3	B3	R2	G2	B2																																																						
<p>All input/output data support 4 endian mode(32bit)</p> <table><tr><td>endian mode</td><td>[31:24] Byte3</td><td>[23:16] Byte2</td><td>[15:8] Byte1</td><td>[7:0] Byte0</td></tr><tr><td>00</td><td>Byte3</td><td>Byte2</td><td>Byte1</td><td>Byte0</td></tr><tr><td>01</td><td>Byte0</td><td>Byte1</td><td>Byte2</td><td>Byte3</td></tr><tr><td>10</td><td>Byte2</td><td>Byte3</td><td>Byte0</td><td>Byte1</td></tr><tr><td>11</td><td>Byte1</td><td>Byte0</td><td>Byte3</td><td>Byte2</td></tr></table> <p>All input/output data support 1 endian mode(64bit)</p> <table><tr><td>endian mode</td><td>[63:32] Byte7-4</td><td>[31:0] Byte3-0</td></tr><tr><td>0</td><td>Byte7-4</td><td>Byte3-0</td></tr><tr><td>1</td><td>Byte3-0</td><td>Byte7-4</td></tr></table> <p>RGB configure</p> <table><tr><td>rgb565 mode</td><td>Sequence</td></tr><tr><td>000</td><td>RGB</td></tr><tr><td>001</td><td>RBG</td></tr><tr><td>010</td><td>GRB</td></tr><tr><td>011</td><td>GBR</td></tr><tr><td>100</td><td>BGR</td></tr><tr><td>101</td><td>BRG</td></tr></table> <p>ARGB 888 configure</p> <table><tr><td>ARGB888 mode</td><td>Sequence</td></tr><tr><td>0</td><td>ARGB</td></tr><tr><td>1</td><td>RGBA</td></tr></table>							endian mode	[31:24] Byte3	[23:16] Byte2	[15:8] Byte1	[7:0] Byte0	00	Byte3	Byte2	Byte1	Byte0	01	Byte0	Byte1	Byte2	Byte3	10	Byte2	Byte3	Byte0	Byte1	11	Byte1	Byte0	Byte3	Byte2	endian mode	[63:32] Byte7-4	[31:0] Byte3-0	0	Byte7-4	Byte3-0	1	Byte3-0	Byte7-4	rgb565 mode	Sequence	000	RGB	001	RBG	010	GRB	011	GBR	100	BGR	101	BRG	ARGB888 mode	Sequence	0	ARGB	1	RGBA
endian mode	[31:24] Byte3	[23:16] Byte2	[15:8] Byte1	[7:0] Byte0																																																								
00	Byte3	Byte2	Byte1	Byte0																																																								
01	Byte0	Byte1	Byte2	Byte3																																																								
10	Byte2	Byte3	Byte0	Byte1																																																								
11	Byte1	Byte0	Byte3	Byte2																																																								
endian mode	[63:32] Byte7-4	[31:0] Byte3-0																																																										
0	Byte7-4	Byte3-0																																																										
1	Byte3-0	Byte7-4																																																										
rgb565 mode	Sequence																																																											
000	RGB																																																											
001	RBG																																																											
010	GRB																																																											
011	GBR																																																											
100	BGR																																																											
101	BRG																																																											
ARGB888 mode	Sequence																																																											
0	ARGB																																																											
1	RGBA																																																											

10.2.5.3 Control Registers

See *UIS7885 DPU VSP System Control Registers Instructions*.

10.2.6 VENC

10.2.6.1 Overview

VPU_ENC is a multi-format video encoder which can perform H.265 and H.264 standard.

10.2.6.2 Features

- Support H.265, main profile, level5.1, 4k@60fps, 1080p@240fps
- Support H.264, baseline profile, main profile, high profile, level 5.1, 4k@24fps 1080p@100fps, 720p@240fps
- Support EIS
- Support YUV 2-Plane/YUV 3-Plane/ARGB8888
- Support AFBC1.2
- Support command queue mode
- Support MB level rate control
- 128bits AXI4.0 master
- 32bits APB3.0 slave

10.2.6.3 Control Registers

See *UIS7885 DPU VSP System Control Registers Instructions*.

10.2.6.4 Application Notes

Some Constraints:

- Source picture must be 16 Bytes alignment in horizontal;
- 3 plane source picture must be 32 Bytes alignment in horizontal result from U/V component must be 16 Bytes alignment in horizontal;
- Don't support 3 plane NV12 result from U/V component need individual pitch;
- Crop needs 4 × 4 block alignment in non-AFBC; 32 × 8 block alignment when AFBC;

10.2.7 VDEC

10.2.7.1 Overview

VPU_DEC is a multi-format video stream processor which can perform decoding of multi standard video streams.

10.2.7.2 Features

- Support H.264, baseline profile, main profile, high profile, high 10 profile, level5.1, 4k@30fps
- Support H.265, main profile, main 10 profile, level5.1, 4k@30fps 10bit, 4k@60fps 8bit
- Support VP9, profile0, profile2, 4k@30fps
- Support VP8, up to 1080p@60fps
- Support AFBC1.2.
- Motion data cache: cache window 128 × 32, cache line 64Byte, 4-way set association.

- 128bits AXI4.0 master.
- 32bits APB3.0 slave.

10.2.7.3 Control Registers

See *UIS7885 DPU VSP System Control Registers Instructions*.

10.2.7.4 Application Notes

Some Constraints:

- Picture should be 32 pixel aligned, and not support 1080i when AFBC mode.
- Must enable rec AFBC mode when apply output scaler.
- Scaler output MUST be 32 pixel aligned.

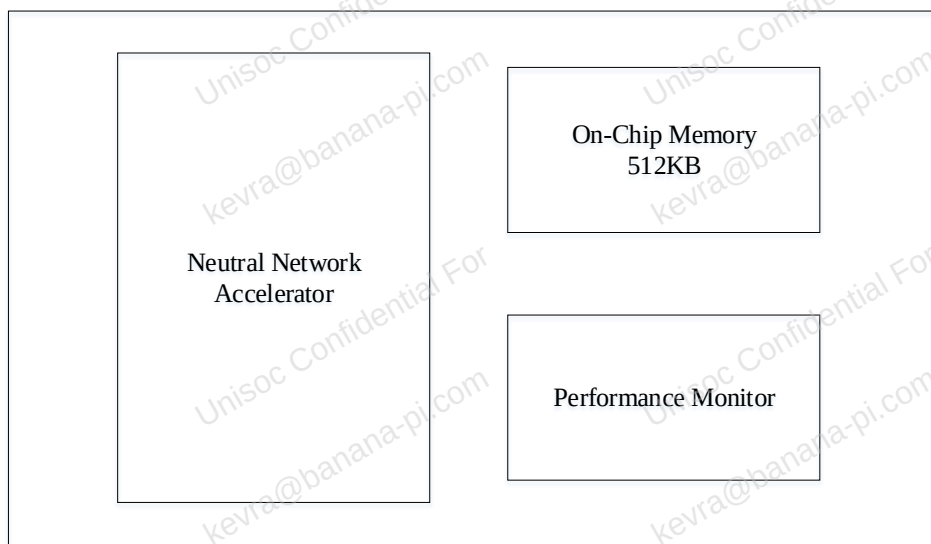
11 AI Subsystem

11.1 System Overview

11.1.1 Functional Block Diagram

AI system consists of several parts listed as below. Major part is neutral network accelerator for processing common neutral network calculation. 512KB on-chip memory provide high performance/bandwidth dedicate memory space for NNA(Neural Network Accelerator). Performance monitor provides indication signal for dynamic frequency voltage scaling by monitoring NNA memory interface, which balance performance and power consumption.

Figure 11-1 AI system architecture



11.1.2 Features

Feature list:

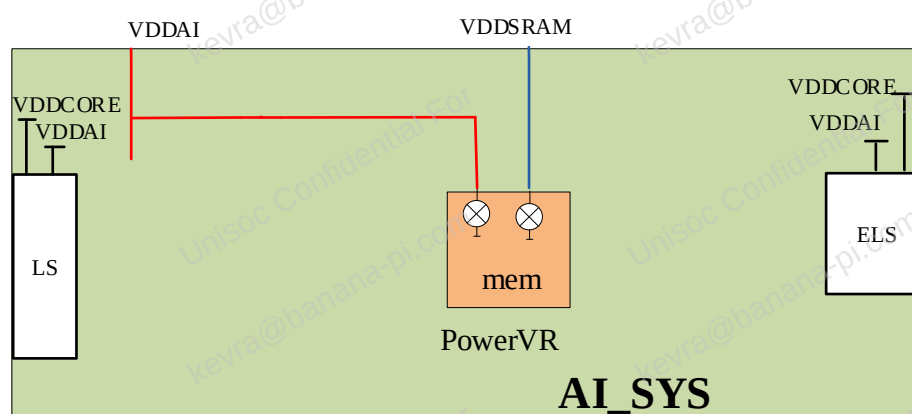
- Neural network accelerator inside
 - Low power, including optimizations when running in low bitdepth modes
 - Support for a range of neural network operations:
 - Convolutions (including atrous and depthwise separable)
 - Activations (ReLU, PReLU, Clamped ReLU, Sigmoid, Tanh)
 - Element-wise operations : Add, Multiply, Max, Min
 - Pooling (Min, Max, Mean)
 - Across-channel Local Response Normalization (LRN)

- Batch Normalization
 - De-convolutions
 - Hardware accelerated memory manipulation operations
- Lossless weight data compression
- DRM security and scheduling for multiple hosts
- Flexible bitdepth support:
 - Support flexible bitdepths on inputs, coefficients, output and intermediate data
 - Supported bitdepths from 4 to 16 bits
- Frameworks and APIs
 - Software tools for offline preparation of networks
 - Adapters for conversion from TensorFlow, TensorFlow Lite, ONNX and Caffe
 - Quantization of floating point networks
 - Offline and runtime network compiler
 - IMGDNN and AndriodNN APIs support
- Bus interface
 - Slave(configuration interface): AXI3 (address width is 32 bits, data width is 32 bits)
 - Master(access DDR): AXI3 (address width is 35 bits, data width is 256 bits)
- Dedicate power supply from DCDC ,improve IR drop degradation
- Rosc for process monitor
- Support djtag debug for key signals
- Support djtag control memory read/write timing window
- Support dvfs: including reactive dvfs and proactive dvfs

11.1.3 System Low power Design

11.1.3.1 Power Scheme

Figure 11-2 AI sys power architecture



- VDDAI voltage supports DVFS.
- VDDSRAM voltage supports 0.75V.

Table 11-1 Power domain in AI SYS

Power Domain Name	Modules
PD_AI_SYS	The whole AI sys

11.1.3.2 Power State

Table 11-2 Power states in AI SYS

Num	Power mode	SYS
1	All off	Off
2	All on	On

11.1.3.3 Low power mode

AI system support low power mode:

- Matrix clock gate
Matrix clock gate is automatically gated when the bus is idle. This is controlled by LPC.
 - When lpc_eb is disable, matrix clock will not be gated.
 - When lpc_eb is enable, lpc is working.
- Light sleep
If all modules that can access ddr are stop in the AI subsystem or busy signal is low, all buffer data to ddr has been transferred, and all lpc for ddr path are in low power state, the subsystem will be in light sleep mode.
- Deep sleep
If all modules in the AI subsystem is idle, the subsystem will be in deep sleep mode.

11.2 Neural network accelerator

11.2.1 Key Features

The key features are :

- Acceleration of the most common layers of neural networks
- AI CORE Acceleration up to 988MHz
- Low bandwidth operation
 - Support for a large range of low precision data formats
 - Grouping layers together to reduce memory bandwidth
 - Lossless weight data compression
- Interoperability:
 - Support for a variety of memory formats designed to share data with a CPU, GPU or other IPs
 - Memory structure of ease of integration into a SoC
 - AXI4 memory bus standard, with support for data reordering and interleaving

- 1TB addressable memory space
- MMU for address translation
- DRM Security and scheduling for multiple hosts

11.2.2 Supported Layers

The NNA is a product designed to perform most operations (layers) of a neural network in the hardware with minimal software interaction. However, some layers are not supported in hardware, and these are run in the software instead. The layers that are handled by software were decided upon by a number of factors including the frequency of the layers usage, the complexity of calculating the layer, and the difficulty of implementing it in hardware

The layers which are supported in hardware are shown below. Further details about these layers and the restrictions which apply are in later sections

Table 11-3 Supported Layers

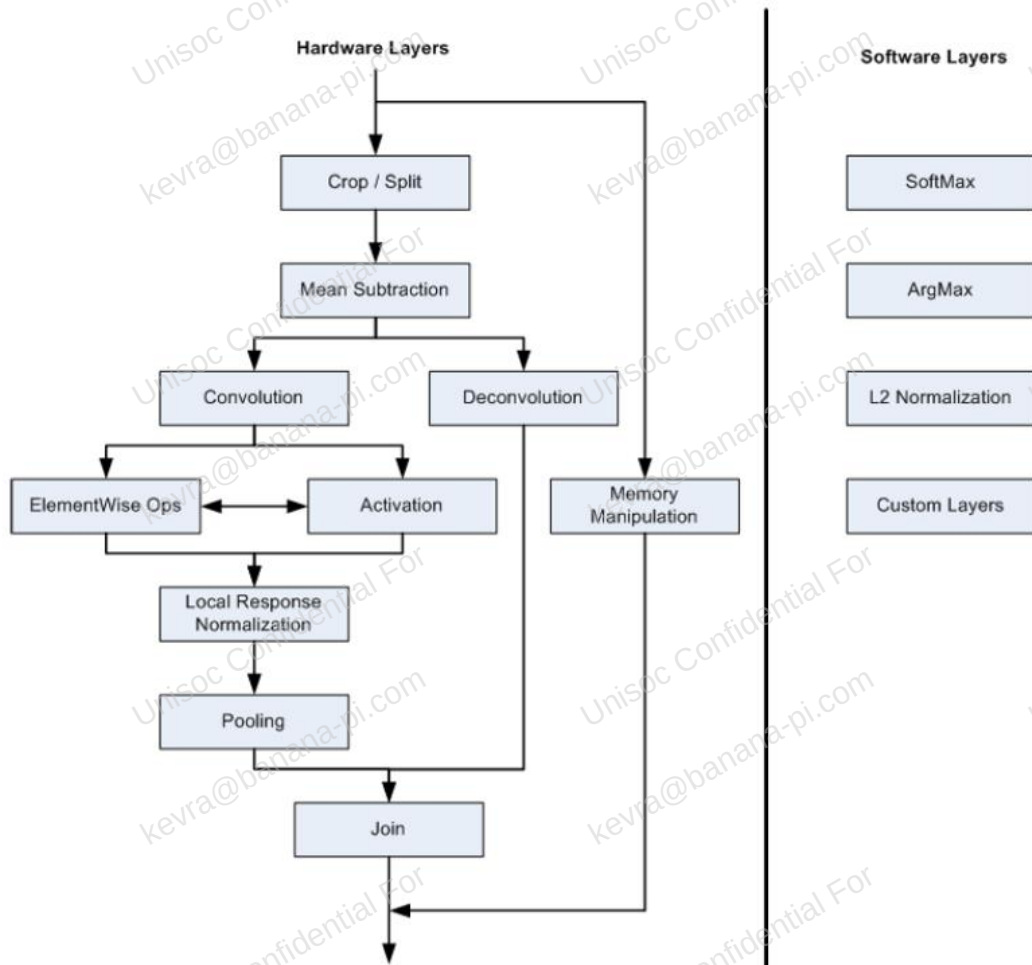
Layer	Features
Convolution	<ul style="list-style-type: none"> • Normal convolution • Atrous/Dilated convolution • Depth wise convolution • Grouped convolution
Fully connected	Fully connected layers
Activation	<ul style="list-style-type: none"> • ReLu • ReLu1 • ReLu6 • Clamped ReLu • Leaky ReLu • PReLu • Logistic/Sigmoid • Tanh • Custom activation functions
Broadcasting	Any dimension combination
Element-Wise Operators	<ul style="list-style-type: none"> • Add/Subtract • Negate • Multiply • Min/Max
Normalisation	<ul style="list-style-type: none"> • Local Response Normalisation (Across Channel) • Batch Normalisation
Pooling	<ul style="list-style-type: none"> • Max polling • Mean polling • Min polling
Pre-Processing	Mean Subtraction

Layer	Features
Depth to Space transformation	
Memory layout	<ul style="list-style-type: none"> • Split • Join • Slice • Flatten • Reshape • Squeeze • Permute/Transpose
Spatial Resize Layers	<ul style="list-style-type: none"> • Pad • Crop • Bi-linear resize • Nearest neighbor resize

11.2.3 Hardware Processing Order

The hardware diagram also shows the order that layers are processed by the hardware, as well as some examples of software layers. The different layers that can be combined together in single pass through the hardware is called a “layer group”. If the order of processing in the target network does not match the order shown in Figure below, then the operations will be spilt into different layer groups. For example to perform local response normalization after pooling the first layer group would perform the pooling layer, and the second would contain the local response normalization layer.

Figure 11-3 Processing Order of Layers in Hardware



12

iSE System Design Specification

12.1 Overview

iSE is an internal embedded security subsystem. It is a microcomputer that can be used together with a chip operating system (COS) to implement functions such as data secure storage, encryption and decryption operations. The iSE system certified by relevant agencies can meet the needs of mobile terminal security key storage, data encryption services, etc. It is widely used in finance, mobile payment, urban transportation and other fields. It can not only protect the security of online payment, but also cooperate with NFC as a wallet for offline payment.

12.2 Features

Isolated execution environment, if need details, please contact UNISOC's support.

13

Reference Documents

1. *UIS7885 Pin List and Pin Register Instructions*
2. *UIS7885 AON System Control Registers Instructions*
3. *UIS7885 AP System Control Registers Instructions*
4. *UIS7885 Audio System Control Registers Instructions*
5. *UIS7885 Camera System Control Registers Instructions*
6. *UIS7885 DPU VSP System Control Registers Instructions.*