Rockchip RK2206ABCD TRM

Revision 1.0
Sep. 2019

Revision History

Date	Revision	Description
2019-9-12	1.0	Initial Release

Table of Content

Table of Content 3
Figure Index 5
Table Index 6
Warranty Disclaimer 7
Chapter 1 Interconnect 8
1.1 Overview 8
1.2 Function Description 8
1.3 Register Description 10
1.4 Application Notes 19
Chapter 2 Process-Voltage-Temperature Monitor (PVTM) 21
2.1 Overview 21
2.2 Block Diagram 21
2.3 Function Description 21
2.5 Application Notes 23
Chapter 3 Crypto 24
3.1 Overview 24
3.2 Block Diagram 24
3.3 Register description 26
3.4 Application Note 78
Chapter 4 Digital Audio Codec 85
4.1 Overview 85
4.2 Block Diagram 86
4.3 Function Description 86
4.4 Register Description 88
4.5 Interface Description 116
4.6 Application Notes 117
Chapter 5 Video Output Processor 119
5.1 Overview 119
5.2 Block Diagram 119
5.3 Function Description 120
5.4 Register Description 122
5.5 Timing Diagram 128
5.6 Application Notes 128
Chapter 6 Video Capture(VICAP) 130
6.1 Overview 130

6.2 Block Diagram 130
6.3 Function Description 130
6.4 Register Description 133
6.5 Interface Description 143
6.6 Application Notes 144
Chapter 7 Mobile Storage Host Controller 145
7.1 Overview 145
7.2 Block Diagram 145
7.3 Function Description 146
7.4 Register Description 168
7.5 Interface Description 194
7.6 Application Notes 194
Chapter 8 USB2.0 OTG 214
8.1 Overview 214
8.2 Block Diagram 214
8.3 USB2.0 OTG Controller 214
8.4 USB2.0 OTG PHY 214
8.5 Register Description 215
8.6 Interface description 215
8.7 Application Note 215
Chapter 9 Audio PWM 216
9.1 Overview 216
9.2 Block Diagram 216
9.3 Function Description 216
9.4 Register Description 217
9.5 Interface Description 223
9.6 Application Notes 223

Figure Index

Fig. 1-1 Idle Request....20
Fig. 2-1 PVTM Block Diagram....21
Fig.3-1 Crypto Architecture....24
Fig.3-2 LLI DMA Usage....80
Fig.3-3 AES-HASH-RX mode....83
Fig.3-4 AES-HASH-TX mode....83
Fig. 4-1 Digital Audio Codec Block Diagram....86
Fig. 5-1 VOP Block Diagram....119
Fig. 5-2 RGB565 data format....120
Fig. 5-3 RGB565 data format....120
Fig. 5-4 YUV420 data format....120
Fig. 5-5 YUV420 data format....120
Fig. 5-6 pixel data path....120
Fig. 5-7 dither down block....121
Fig. 5-8 i8080 r/w timing....128
Fig. 5-9 no split....128
Fig. 5-10 two-phase split....128
Fig. 5-11 four-phase split....128
Fig. 5-12 operation flow....129
Fig. 6-1 VICAP Block Diagram....130
Fig. 6-2 Timing diagram for VICAP when vsync low active....131
Fig. 6-3 Timing diagram for VICAP when vsync high active....131
Fig. 6-4 Timing diagram for VICAP when href high active....131
Fig. 6-5 Timing diagram for VICAP when href low active....131
Fig. 6-6 Timing diagram for VICAP when Y data first....131
Fig. 6-7 Timing diagram for VICAP when U data first....131
Fig. 6-8 CCIR656 timing....132
Fig. 6-9 Raw Data or JPEG Timing....132
Fig. 7-1 Host Controller Block Diagram....146
Fig. 7-2 SD/MMC Card-Detect Signal....151
Fig. 7-3 Host Controller Command Path State Machine....153
Fig. 7-4 Host Controller Data Transmit State Machine....155
Fig. 7-5 Host Controller Data Receive State Machine....157
Fig. 7-6 Dual-Buffer Descriptor Structure....163
Fig. 7-7 Chain Descriptor Structure....163
Fig. 7-8 Descriptor Formats for 32-bit AHB Address Bus Width....164
Fig. 7-9 SD/MMC Card Termination....195
Fig. 7-10 Host Controller Initialization Sequence....197
Fig. 7-11 Voltage Switching Command Flow Diagram....207
Fig. 7-12 ACMD41 Argument....207
Fig. 7-13 ACMD41 Response(R3)....208
Fig. 7-14 Voltage Switch Normal Scenario....208
Fig. 7-15 Voltage Switch Error Scenario....209
Fig. 7-16 Clock Generation Unit....212
Fig. 9-1 Block Diagram of Audio PWM....216

Table Index

Table 1-1 QoS Generator 9
Table 1-2 Probe 9
Table 3-1 Crypto Clock & Reset Description 78
Table 3-2 Crypto Clock & Reset Description 79
Table 3-3 Crypto Performance Description 79
Table 3-4 LLI Item Description 80
Table 3-5 LLI Item dma_ctl Description 80
Table 3-6 LLI Item user_define Description 81
Table 3-7 LLI Item user_define Description 81
Table 3-8 LLI Item user_define Description 82
Table 4-1 Equivalent parameters of digital ADC filters 87
Table 4-2 Digital Audio Codec Group 0 Interface Description 116
Table 4-3 Digital Audio Codec Group 1 Interface Description 116
Table 4-4 Relationship of ACDC_CLK, D2A_CLK, D2A_SYNC and Sample Rates in Normal Mode 117
Table 4-5 Relationship of ACDC_CLK, D2A_CLK, D2A_SYNC and sample rates in low power mode 1 117
Table 4-6 Relationship of ACDC_CLK, D2A_CLK, D2A_SYNC and sample rates in low power mode 2 117
Table 5-1 VOP output pins 121
Table 6-1 VICAP Interface Description 143
Table 7-1 Bits in Interrupt Status Register 148
Table 7-2 Auto-Stop Generation 158
Table 7-3 Non-data Transfer Commands and Requirements 160
Table 7-4 Bits in IDMAC DES0 Element 164
Table 7-5 Bits in IDMAC DES1 Element 165
Table 7-6 Bits in IDMAC DES2 Element 165
Table 7-7 Bits in IDMAC DES3 Element 165
Table 7-8 SDMMC Interface Description 194
Table 7-9 Recommended Usage of use_hold_reg 196
Table 7-10 Command Settings for No-Data Command 200
Table 7-11 Command Setting for Single or Multiple-Block Read 201
Table 7-12 Command Settings for Single or Multiple-Block Write 203
Table 7-13 PBL and Watermark Levels 211
Table 7-14 Configuration for SDMMC Clock Generation 212
Table 8-1USB2.0 OTG Interface Description 215
Table 9-1 Duty Cycle Resolution of the Audio PWM 216
Table 9-2 Audio PWM Interface Description 223

Warranty Disclaimer

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

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

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

Copyright and Patent Right

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

Rockchip Electronics Co.,Ltd does not convey any license under its copyright and patent rights nor the rights of others.
All copyright and patent rights referenced in this document belong to their respective owners and shall be subject to corresponding copyright and patent licensing requirements.

Trademarks

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

Confidentiality

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

Reverse engineering or disassembly is prohibited.

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

Copyright © 2019 Rockchip Electronics Co.,Ltd.

Chapter 1 Interconnect

1.1 Overview

The chip-level interconnect consists of main interconnect. It enables communication among the modules and subsystems in the device.
The main interconnect supports the following features:

- Cross-bar exchange network
- A special internal slave for accessing the configuration register
- Little-endian platform
- QoS management for optimizing the transaction flow
- Transaction statistics for analyzing the transaction flow

1.2 Function Description

1.2.1 Interconnect Services

The Interconnect supports run-time tuning, via control registers, and observed via status registers. Interconnect service networks provide access to registers.
The registers include the QosGenerator, and Probe registers.

1.2.2 QoS(Quality of Service) management

The main interconnect is connected with all the related IPs of the system, the interface between the IP and the interconnect is called as NIU(native interface unit).
The interconnect offers 4 modes of QoS management:

- None, QoSGenerator is disabled, and priority information are stuck at 0.
- Fixed, QoSGenerator drives apply a fixed urgency to read transactions, and a (possibly different) urgency to write transactions.
- Limiter, QoSGenerator behaves as in fixed mode, but limits the traffic bandwidth coming from that socket, possibly stalling requests if the initiator attempts to exceed its budget.
- Regulator, QoSGenerator promotes are demotes hurry, depending the bandwidth obtained by the initiator is below or beyond a bandwidth budget. As transactions exceeding the bandwidth limit are sent (even though demoted), the regulator mode may be considered as a softer version of the limiter mode.

Limiter Behavior

When configured in bandwidth limiter, the unit uses a 23 bit counter to measure the average bandwidth. This counter has a 1/256 byte resolution and works as follows:

- Adds the number of byte rounded up to 16 (1 -> 16) and then multiplied by 256 to the current value, each time a request is sent.
- Subtracts the Bandwidth register value every cycle. If the Counter becomes negative, force it to 0.
- If the Counter value is greater than the Saturation register value multiplied by 16*256, any incoming request is stalled until this condition disappears. Note that the Counter cannot wrap-around because the maximum value it can reach is: SaturationMax*16*256 + BurstMax*256 = 1023*4K+4K*256 = 5116K or 223 = 8192K.

The following example will show the Counter behavior: 32 byte bursts, F=400MHz, BW=200MB/s, T=0.32us. The Bandwidth register will be set to 256*200/400 = 128, and the Saturation register to 128*0.32*400/4096 = 4 (which corresponds to 64 bytes).

Regulator Behavior

When configured in bandwidth regulator, the unit uses a 23 bit counter to measure the average bandwidth. This counter has a 1/256 byte resolution and works as follows:

- Adds the number of byte rounded up to 16 and then multiplied by 256 to the current value, each time a response is received. If the result is greater than the Saturation register value multiplied by 16*256, saturation to this value is applied.
- Subtracts the Bandwidth register value every cycle. If the Counter becomes negative,

force it to 0.

- If the Counter value is less than or equal to the Saturation register value multiplied by 16*256/2, the SocketMst Hurry signal will be set to the HurryHigh register, and HurryLow otherwise. Note that Urgency and Press will be also set to the same value.

The following example will show the Counter behavior: 1Kbyte bursts, F=500MHz, BW=2GB/s,T=2.048us. The Bandwidth register will be set to 256*2000/500 = 1024, and the Saturation register to 1024*2.048*500/4096 = 256 (which corresponds to 4 Kbytes).

QoS Generator Programming

Bandwidth: This log2 (socket.wData/8)+8 bits register defines the bandwidth in 1/256th byte per cycle unit. This allows a 2 MByte/s resolution at 500MHz. When the bandwidth is given in MByte/s, the value of this register will be equal to 256*BWMB/s / FMHz.

Saturation: This 10 bits register defines the number of byte used for bandwidth measurement. It is expressed in 16bytes unit (up to 16 Kbyte). Usually the integration window is given in us or in cycle: the value of this register will be equal to Bandwidth*Tus*FMHz/(256*16) or Bandwidth*Ncycle/(256*16).

The QoS management for peri-interconnect and main-interconnect inside each interconnect is independent. When master in peri-interconnect access main-interconnect, the QoS will be propagated to main-interconnect according to QoS configuration for 'peri2msch_nsp' (peri master access DDR). The masters in main-interconnect does not access peri-interconnect.

The default setting of master NIUs are listing below:

Table 1-1 QoS Generator

Master NIU	Priority0/1	Register Base Address
dma	3	0x40290400
dsp	3	0x40290480
M0	3	0x40290500
M4ID	3	0x40290600
M4S	3	0x40290680
sfc	3	0x40290700
crypto	3	0x40291400
sdmmc	3	0x40291480
spi2apb	3	0x40291500
usb	3	0x40291580
vicap	3	0x40291600
wlan	3	0x40292400

Note:

- *All master NIU QoS generator mode is 'Regulator', bandwidth is 100MB/s and saturation is 1024.*
- *The bandwidth must be calculated based on actual operating frequency.*
- *Refer to chapter 11.3 for detail register. All generators have the same register except the valid bits of 'Bandwidth' filed may be different.*

1.2.3 Probe

The interconnect provides a service called probe to trace packet and compute traffic statics, there are totally 3 probes to monitor the memory traffic statics and each can be programmed by their register. They are listed below.

Table 1-2 Probe

Probe Name	Monitor Path	Register Base Address
mcu_probe	pd_mcu access internal memory 0/1	0x40290000
peri_probe	pd_peri access pd_mcu	0x40291000
wlan_probe	pd_wlan access pd_mcu	0x40292000

Refer to chapter 11.3 for detail register.

1.3 Register Description

1.3.1 Internal Address Mapping

Name	Offset	Description
mcu_Probe	0x0000	service_mcu + offset
dma_QosGenerator	0x0400	service_mcu + offset
dsp_QosGenerator	0x0480	service_mcu + offset
m0_QosGenerator	0x0500	service_mcu + offset
m4id_QosGenerator	0x0600	service_mcu + offset
m4s_QosGenerator	0x0680	service_mcu + offset
sfc_QosGenerator	0x0700	service_mcu + offset
obsrv_mcu	0x0800	service_mcu + offset
peri_Probe	0x0000	service_peri + offset
crypto_QosGenerator	0x0400	service_peri + offset
sdmmc_QosGenerator	0x0480	service_peri + offset
spi2apb_QosGenerator	0x0500	service_peri + offset
usb_QosGenerator	0x0580	service_peri + offset
vicap_QosGenerator	0x0600	service_peri + offset
sdmmc_QosGenerator	0x0280	service_peri + offset
sfc_QosGenerator	0x0300	service_peri + offset
obsrv_peri	0x0800	service_peri + offset
wlan_Probe	0x0000	service_wlan + offset
wlan_QosGenerator	0x0400	service_wlan + offset
obsrv_wlan	0x0800	service_wlan + offset

1.3.2 QoS Registers Summary

Name	Offset	Size	Reset Value	Description
QOS_Id_CoreId	0x0000	W	0xeba52304	Core ID
QOS_Id_RevisionId	0x0004	W	0x00018100	Revision ID
QOS_Priority	0x0008	W	0x80000202	QoS prirority
QOS_Mode	0x000c	W	0x00000003	Qos mode seletion
QOS_Bandwidth	0x0010	W	0x00000140	QoS bandwidth
QOS_Saturation	0x0014	W	0x00000040	QoS saturation
QOS_ExtControl	0x0018	W	0x00000000	Qos external control

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

1.3.3 QoS Detail Register Description

QOS_Id_CoreId

Address: Operational Base + offset (0x0000)

Bit	Attr	Reset Value	Description
31:8	RO	0xeba523	CoreChecksum Field containing a checksum of the parameters of the IP
7:0	RO	0x04	CoreTypeId Field identifying the type of IP

QOS_Id_RevisionId

Address: Operational Base + offset (0x0004)

Bit	Attr	Reset Value	Description
31:0	RO	0x00018100	RevisionId Constant

QOS_Priority

Address: Operational Base + offset (0x0008)

Bit	Attr	Reset Value	Description
31	RO	0x1	Mark Backward compatibility marker when 0
30:10	RO	0x0	reserved
9:8	RW	0x2	P1 In Programmable or Bandwidth Limiter mode, the priority level for read transactions. In Bandwidth regulator mode, the priority level when the used throughput is below the threshold. In Bandwidth Regulator mode, P1 should have a value equal or greater than P0.
7:2	RO	0x0	reserved
1:0	RW	0x2	P0 In Programmable or Bandwidth Limiter mode, the priority level for write transactions. In Bandwidth Regulator mode, the priority level when the used throughput is above the threshold. In Bandwidth Regulator mode, P0 should have a value equal or lower than P1.

QOS_Mode

Address: Operational Base + offset (0x000c)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1:0	RW	0x3	Mode 0 = Programmable mode: a programmed priority is assigned to each read or write, 1 = Bandwidth Limiter Mode: a hard limit restricts throughput, 2 = Bypass mode: (<See SoC-specific QoS generator documentation>), 3 = Bandwidth Regulator mode: priority decreases when throughput exceeds a threshold.

QOS_Bandwidth

Address: Operational Base + offset (0x0010)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:0	RW	0x140	Bandwidth In Bandwidth Limiter or Bandwidth Regulator mode, the bandwidth threshold in units of 1/256th bytes per cycle. For example, 80 MBps on a 250 MHz interface is value 0x0052. The valid bits may be different for different master NIU.

QOS_Saturation

Address: Operational Base + offset (0x0014)

Bit	Attr	Reset Value	Description
31:10	RO	0x0	reserved
9:0	RW	0x040	Saturation In Bandwidth Limiter or Bandwidth Regulator mode, the maximum data count value, in units of 16 bytes. This determines the window of time over which bandwidth is measured. For example, to measure bandwidth within a 1000 cycle window on a 64-bit interface is value 0x1F4.

QOS_ExtControl

Address: Operational Base + offset (0x0018)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	SocketQosEn Register field SocketQosEn determines how priority levels are driven when QoS generators and socket interfaces alternatively drive the levels for Urgency, Pressure, and Hurry signals: When set to 0, the QoS generator drives the levels. When set to 1, internal signals Pressure and Hurry are driven by the greater of the two levels from the socket interface or the QoS generator.

1.3.4 Probe Registers Summary

Name	Offset	Size	Reset Value	Description
PROBE_Id_CoreId	0x0000	W	0x54af1e06	Core ID
PROBE_Id_RevisionId	0x0004	W	0x00018100	Revision ID
PROBE_MainCtl	0x0008	W	0x00000000	Register MainCtl contains probe global control bits
PROBE_CfgCtl	0x000c	W	0x00000000	Register CfgCtl contains global enable and active bits. The register, which must be used by software before changing certain packet probe global registers
PROBE_StatPeriod	0x0024	W	0x00000000	Statistics Period
PROBE_StatGo	0x0028	W	0x00000000	Statistics begin control
PROBE_Counters_0_Src	0x0138	W	0x00000000	Register CntSrc indicates the event source used to increment the counter. Unassigned values (non-existing Press level or ExtEvent index, or unimplemented Filter) are equivalent to OFF
PROBE_Counters_0_Val	0x013c	W	0x00000000	Registers Counters_M_Val contain the statistics counter values

Name	Offset	Size	Reset Value	Description
PROBE_Counters_1_Src	0x014c	W	0x00000000	Register CntSrc indicates the event source used to increment the counter. Unassigned values (non-existing Press level or ExtEvent index, or unimplemented Filter) are equivalent to OFF
PROBE_Counters_1_Val	0x0150	W	0x00000000	Registers Counters_M_Val contain the statistics counter values
PROBE_Counters_2_Src	0x0160	W	0x00000000	Register CntSrc indicates the event source used to increment the counter. Unassigned values (non-existing Press level or ExtEvent index, or unimplemented Filter) are equivalent to OFF
PROBE_Counters_2_Val	0x0164	W	0x00000000	Registers Counters_M_Val contain the statistics counter values
PROBE_Counters_3_Src	0x0174	W	0x00000000	Register CntSrc indicates the event source used to increment the counter. Unassigned values (non-existing Press level or ExtEvent index, or unimplemented Filter) are equivalent to OFF
PROBE_Counters_3_Val	0x0178	W	0x00000000	Registers Counters_M_Val contain the statistics counter values

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

1.3.5 Probe Detail Register Description

PROBE_Id_CoreId

Address: Operational Base + offset (0x0000)

Bit	Attr	Reset Value	Description
31:8	RO	0x54af1e	CoreChecksum Field containing a checksum of the parameters of the IP
7:0	RO	0x06	CoreTypeId Field identifying the type of IP

PROBE_Id_RevisionId

Address: Operational Base + offset (0x0004)

Bit	Attr	Reset Value	Description
31:0	RO	0x00018100	RevisionId Constant

PROBE_MainCtl

Address: Operational Base + offset (0x0008)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	FiltByteAlwaysChainableEn When set to 0, filters are mapped to all statistic counters when counting bytes or enabled bytes. Therefore, only filter events mapped to even counters can be counted using a pair of chained counters.When set to 1, filters are mapped only to even statistic counters when counting bytes or enabled bytes. Thus events from any filter can be counted using a pair of chained counters.
6	RO	0x0	IntrusiveMode When set to 1, register field IntrusiveMode enables trace operation in Intrusive flow-control mode. When set to 0, the register enables trace operation in Overflow flow-control mode.
5	RW	0x0	StatCondDump When set, register field StatCondDump enables the dump of a statistics frame to the range of counter values set for registers StatAlarmMin, StatAlarmMax, and AlarmMode. This field also renders register StatAlarmStatus inoperative. When parameter statisticsCounterAlarm is set to False, the StatCondDump register bit is reserved.
4	RW	0x0	AlarmEn When set, register field AlarmEn enables the probe to collect alarm-related information. When the register field bit is null, both TraceAlarm and StatAlarm outputs are driven to 0.
3	RW	0x0	StatEn When set to 1, register field StatEn enables statistics profiling. The probe sendS statistics results to the output for signal ObsTx. All statistics counters are cleared when the StatEn bit goes from 0 to 1. When set to 0, counters are disabled.
2	RW	0x0	PayloadEn Register field PayloadEn, when set to 1, enables traces to contain headers and payload. When set ot 0, only headers are reported.
1	RO	0x0	TraceEn Register field TraceEn enables the probe to send filtered packets (Trace) on the ObsTx observation output.
0	RW	0x0	ErrEn Register field ErrEn enables the probe to send on the ObsTx output any packet with Error status, independently of filtering mechanisms, thus constituting a simple supplementary global filter.

PROBE_CfgCtl

Address: Operational Base + offset (0x000c)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved

Bit	Attr	Reset Value	Description
1	RO	0x0	Active Register field Active is used to inform software that the probe is active. Probe configuration is not allowed during the active state. This bit is raised when bit GlobalEn is set, and is cleared a few cycles after setting GlobalEn to zero (probe is Idle).
0	RW	0x0	GlobalEn Set register field GlobalEn to 1 enable the tracing and statistics collection sub-systems of the packet probe.

PROBE_StatPeriod

Address: Operational Base + offset (0x0024)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:0	RW	0x00	StatPeriod Register StatPeriod is a 5-bit register that sets a period, within a range of 2 cycles to 2 gigacycles, during which statistics are collected before being dumped automatically. Setting the register implicitly enables automatic mode operation for statistics collection. The period is calculated with the formula: N_Cycle = 2**StatPeriodWhen register StatPeriod is set to its default value 0, automatic dump mode is disabled, and register StatGo is activated for manual mode operation. Note: When parameter statisticsCollection is set to False, Stat Period is reserved.

PROBE_StatGo

Address: Operational Base + offset (0x0028)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	StatGo Writing a 1 to the 1-bit pulse register StatGo generates a statistics dump. The register is active when statistics collection operates in manual mode, that is, when register StatPeriod is set to 0. NOTE The written value is not stored in StatGo. A read always returns 0.

PROBE_Counters_0_Src

Address: Operational Base + offset (0x0138)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved

Bit	Attr	Reset Value	Description
4:0	RW	0x00	IntEvent Internal packet event 5'h00 OFF Counter disabled. 5'h01 CYCLE8 Probe clock cycles. 5'h02 IDLE Idle cycles during which no packet data is observed. 5'h03 XFER Transfer cycles during which packet data is transferred. 5'h04 BUSY Busy cycles during which the packet data is made available by the transmitting agent but the receiving agent is not ready to receive it. 5'h05 WAIT Wait cycles during a packet in which the transmitting agent suspends the transfer of packet data. 5'h06 PKT Packets. 5'h08 BYTE Total number of payload bytes. 5'h09 PRESS Clock cycles with pressure level > 0. 5'h0A PRESS Clock cycles with pressure level > 1. 5'h0B PRESS Clock cycles with pressure level > 2. 5'h10 CHAIN Carry from counter 2m to counter 2m + 1

PROBE_Counters_0_Val

Address: Operational Base + offset (0x013c)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RO	0x0000	Counters_0_Val Register Val is a read-only register that is always present. The register contains the statistics counter value either pending StatAlarm output, or when statistics collection is suspended subsequent to triggers or signal stat Suspend.

PROBE_Counters_1_Src

Address: Operational Base + offset (0x014c)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved

Bit	Attr	Reset Value	Description
4:0	RW	0x00	IntEvent Internal packet event Event | source type | Event description 5'h00 OFF Counter disabled. 5'h01 CYCLE8 Probe clock cycles. 5'h02 IDLE Idle cycles during which no packet data is observed. 5'h03 XFER Transfer cycles during which packet data is transferred. 5'h04 BUSY Busy cycles during which the packet data is made available by the transmitting agent but the receiving agent is not ready to receive it. 5'h05 WAIT Wait cycles during a packet in which the transmitting agent suspends the transfer of packet data. 5'h06 PKT Packets. 5'h08 BYTE Total number of payload bytes. 5'h09 PRESS Clock cycles with pressure level > 0. 5'h0A PRESS Clock cycles with pressure level > 1. 5'h0B PRESS Clock cycles with pressure level > 2. 5'h10 CHAIN Carry from counter 2m to counter 2m + 1

PROBE_Counters_1_Val

Address: Operational Base + offset (0x0150)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RO	0x0000	Counters_0_Val Register Val is a read-only register that is always present. The register contains the statistics counter value either pending StatAlarm output, or when statistics collection is suspended subsequent to triggers or signal stat Suspend.

PROBE_Counters_2_Src

Address: Operational Base + offset (0x0160)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved

Bit	Attr	Reset Value	Description
4:0	RW	0x00	IntEvent Internal packet event Event | source type | Event description 5'h00 OFF Counter disabled. 5'h01 CYCLE8 Probe clock cycles. 5'h02 IDLE Idle cycles during which no packet data is observed. 5'h03 XFER Transfer cycles during which packet data is transferred. 5'h04 BUSY Busy cycles during which the packet data is made available by the transmitting agent but the receiving agent is not ready to receive it. 5'h05 WAIT Wait cycles during a packet in which the transmitting agent suspends the transfer of packet data. 5'h06 PKT Packets. 5'h08 BYTE Total number of payload bytes. 5'h09 PRESS Clock cycles with pressure level > 0. 5'h0A PRESS Clock cycles with pressure level > 1. 5'h0B PRESS Clock cycles with pressure level > 2. 5'h10 CHAIN Carry from counter 2m to counter 2m + 1

PROBE_Counters_2_Val

Address: Operational Base + offset (0x0164)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RO	0x0000	Counters_0_Val Register Val is a read-only register that is always present. The register contains the statistics counter value either pending StatAlarm output, or when statistics collection is suspended subsequent to triggers or signal stat Suspend.

PROBE_Counters_3_Src

Address: Operational Base + offset (0x0174)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved

Bit	Attr	Reset Value	Description
4:0	RW	0x00	IntEvent Internal packet event Event | source type | Event description 5'h00 OFF Counter disabled. 5'h01 CYCLE8 Probe clock cycles. 5'h02 IDLE Idle cycles during which no packet data is observed. 5'h03 XFER Transfer cycles during which packet data is transferred. 5'h04 BUSY Busy cycles during which the packet data is made available by the transmitting agent but the receiving agent is not ready to receive it. 5'h05 WAIT Wait cycles during a packet in which the transmitting agent suspends the transfer of packet data. 5'h06 PKT Packets. 5'h08 BYTE Total number of payload bytes. 5'h09 PRESS Clock cycles with pressure level > 0. 5'h0A PRESS Clock cycles with pressure level > 1. 5'h0B PRESS Clock cycles with pressure level > 2. 5'h10 CHAIN Carry from counter 2m to counter 2m + 1

PROBE_Counters_3_Val

Address: Operational Base + offset (0x0178)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RO	0x0000	Counters_0_Val Register Val is a read-only register that is always present. The register contains the statistics counter value either pending StatAlarm output, or when statistics collection is suspended subsequent to triggers or signal stat Suspend.

1.4 Application Notes

1.4.1 QoS setting

The VOP has the external QoS control.

It's recommended that field 0 of QoS. ExtControl set to 1 to enable the external QoS control. And priority setting of each master kept at 1.

1.4.2 Idle request

The main interconnect supports flushing the ongoing transaction when the software needed to do so.

If the power domain need to disconnect from the main interconnect, Idle request has to be sent to NIU, the NIU will respond a ack, and when it's ready to be disconnect, one Idle signal will be send out . Then, if the power domain still have transaction to be sent to the memory scheduler, it will be stalled by the NIU.

If the power domain is disconnected as the above flow, then CPU want to access to the power domain, it will response error or hold to CPU according to the corresponding grf register setting.

The sequence is like following figure shows:

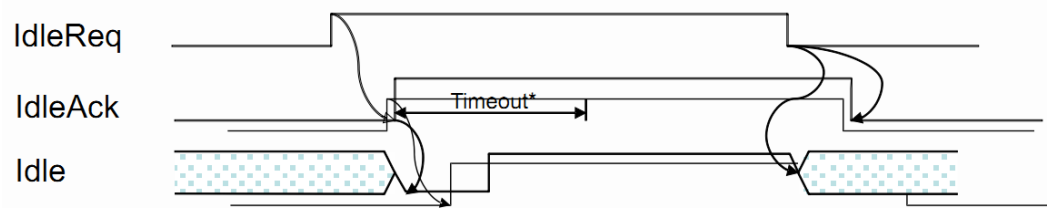

Fig. 1-1 Idle Request

The idle request is set by PMU register.

1.4.3 Basic Packet Tracing

To trace packets, the packet probe must be programmed as follows:

- Select the interesting probe listed in section 11.2.3.
- Set field *TraceEn* of register *MainCtl* to 1 to enable forwarding of traced packets to the connected observer. Optionally set field *PayloadEn* of register *MainCtl* to 1 if the packet payload should be included in the trace.
- Set field *GlobalEn* of register *CfgCtl* to 1.

1.4.4 Counting packets over a fixed period

The following programming sequence counts packets at a given probe point using statistic counter 0.

- Select the interesting probe listed in section 11.2.3
- Set field *StatEn* to 1 in register *MainCtl*.
- Set register *Counters_0_Src* to 0x6 (PKT)to count packets.
- Specify the period during which the packets should be counted by setting register StatPeriod to: $\log_2$(interval expressed in number of probe clock cycles).
- Set field *GlobalEn* of register *CfgCtl* to 1 to enable packet counting.

Once time $2^{StatPeriod}$ has elapsed, the number of packets counted is dumped to the observer and can be read from *Counters_0_Val*.

1.4.5 Measuring bandwidth

The following programming sequence example shows how a packet probe can be used to measure bandwidth at a probe point.

Some important points to note about this example are:

- Statistics counters are chained together to support the maximum theoretical bandwidth. Counter 0 is configured to count bytes; counter 1 increments when counter 0 rolls over.
- The counter values are dumped to an observer after time $2^{StatPeriod}$.

The programming sequence is as follows:

- Select the interesting probe listed in section 11.2.3.
- Set register *Counters_0_Src* to 0x8 (BYTES) to count bytes.
- Set register *Counters_1_Src* to 0x10 (CHAIN) to increment when counter 0 wraps.
- Specify the period during which the bytes should be counted by setting register *StatPeriod* to: $\log_2$(interval expressed in number of probe clock cycles).
- Set field *GlobalEn* of register *CfgCtl* to 1 to enable the counting of bytes.

Once time $2^{StatPeriod}$ has elapsed, the number of packets counted is dumped to the observer and can be read from *Counters_0_Val* and *Counters_0_Val*.

Chapter 2 Process-Voltage-Temperature Monitor (PVTM)

2.1 Overview

The Process-Voltage-Temperature Monitor (PVTM) is used to monitor the chip performance variance caused by chip process, voltage and temperature.
PVTM supports the following features:

- A clock oscillation ring is integrated and used to generate a clock like signal, the frequency of this clock is determined by the cell delay value of clock oscillation ring circuit.
- A frequency counter is used to measure the frequency of the clock oscillator ring.

2.2 Block Diagram

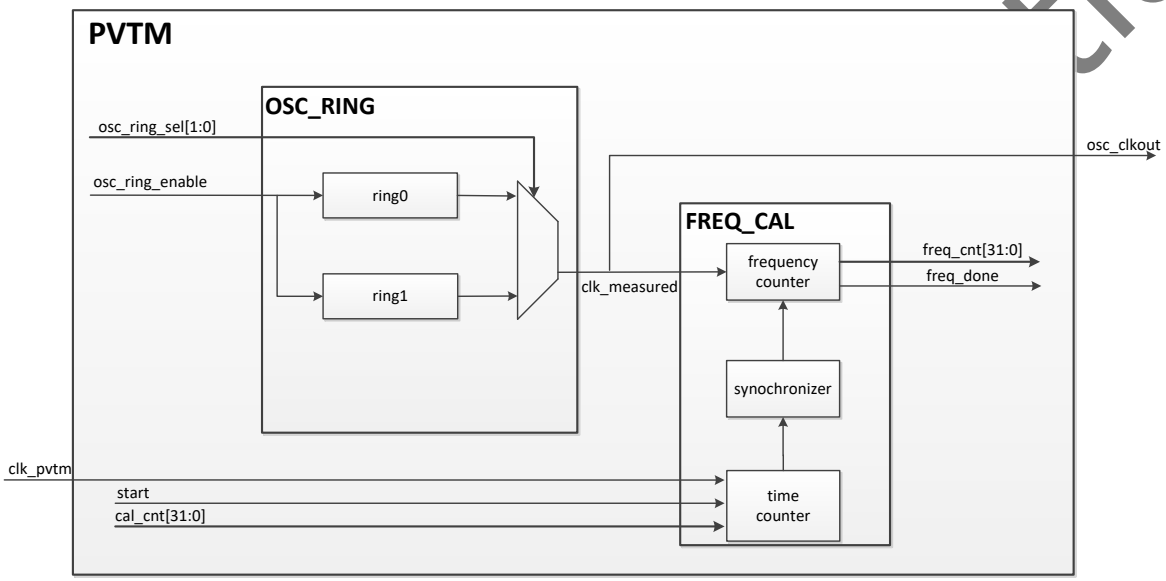

Fig. 2-1 PVTM Block Diagram

The PVTM include two main blocks:

- OSC_RING

It is composed with inverters with odd number, which is used to generate a clock. It supports 2 clock oscillator rings, and finally selects one of them as output.

- FREQ_CAL

It is used to measure the frequency of clock which generated from the OSC_RING block.

2.3 Function Description

2.3.1 Frequency Calculation

A clock is generated by the OSC_RING, and a frequency fixed clock clk_pvtm (40MHz) is used to calculate the cycles of the clock. Supposing the time period is 1s, then the clock period of OSC_RING clock is T= 1/clock_counter(s), the cell delay value is T/2.

2.3.2 Low power mode usage

A clock divided from PVTM oscillation ring is used in low power mode, which can replace the function of 32KHz clock source by configure CRU_CLKSEL_CON00[7:5]. The division factor is configured by CRU_CLKSEL_CON01.

2.4 Register Description

2.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
PVTM_VERSION	0x0000	W	0x00000200	PVTM version register

Name	Offset	Size	Reset Value	Description
PVTM_CON0	0x0004	W	0x00000000	PVTM control register0
PVTM_CON1	0x0008	W	0x00000000	PVTM control register1
PVTM_STATUS0	0x0080	W	0x00000000	PVTM status register0
PVTM_STATUS1	0x0084	W	0x00000000	PVTM status register1

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

2.4.2 Detail Register Description

PVTM_VERSION

Address: Operational Base + offset (0x0000)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RW	0x0200	version PVTM version

PVTM_CON0

Address: Operational Base + offset (0x0004)

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable Write enable for lower 16 bits, each bit is individual. 1'b0: Write access disable 1'b1: Write access enable
15:4	RO	0x0	reserved
3:2	RW	0x0	pvtm_osc_sel Oscillator ring selection 2'b00: oscillator ring 0 2'b01: oscillator ring 1 Others: Reserved
1	RW	0x0	pvtm_osc_en Set high to enable the OSC_RING in the PVTM.
0	RW	0x0	pvtm_start Set high to start PVTM.

PVTM_CON1

Address: Operational Base + offset (0x0008)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	pvtm_cal_cnt PVTM calculation counter

PVTM_STATUS0

Address: Operational Base + offset (0x0080)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	pvtm_freq_done Indicates PVTM frequency count done.

PVTM_STATUS1

Address: Operational Base + offset (0x0084)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	pvtm_freq_cnt Indicates the cycle counts of the OSC_RING clock.

2.5 Application Notes

2.5.1 PVTM Usage Flow

1. Enable the frequency fixed clock clk_pvtm.
2. Reset the PVTM.
3. Set osc_ring_enable '1' to enable the generated clock.
4. Set osc_ring_sel to select the clock oscillation ring
5. Configure the cal_cnt to an appropriate value.
6. Set start '1' to calculate the cycles of the generated clock.
7. Wait the freq_done is asserted, then get the value of freq_cnt. The period OSC_RING clock is T= cal_cnt*(Period of 40MHz clock)/freq_cnt, the cell delay value is T/2.

Chapter 3 Crypto

3.1 Overview

Crypto is a hardware accelerator for encrypting or decrypting. It supports the most commonly used algorithm: DES/3DES, AES, SHA1, SHA256, MD5 and PKA.
The Crypto supports following features:

- Support Link List Item (LLI) DMA transfer;
- Support SHA-1, SHA-256/224, SHA-512/384, MD5 with hardware padding;
- Support HMAC ofSHA-1, SHA-256, SHA-512, MD5 with hardware padding;
- Support AES-128, AES-192, AES-256 encrypt & decrypt cipher;
- Support DES & TDES cipher;
- Support AES ECB/CBC/OFB/CFB/CTR/CTS/XTS/CCM/GCM/CBC-MAC/CMAC mode;
- Support DES/TDES ECB/CBC/OFB/CFB mode;
- Support up to 4096bits PKA mathematical operations for RSA/ECC;
- Support Up to 256 bits TRNG Output.

3.2 Block Diagram

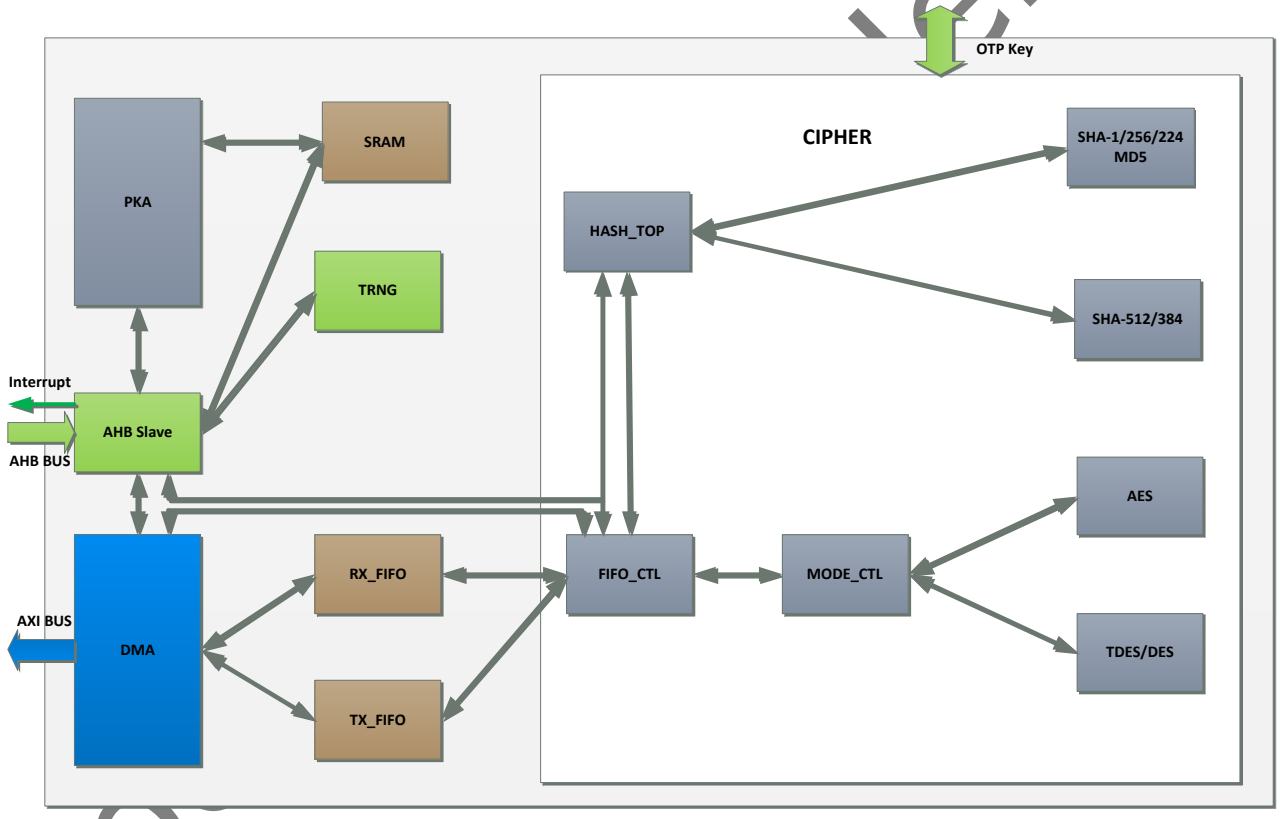

Fig.3-1 Crypto Architecture

Crypto contains several modules: AHB_Slave, DMA, CIPHER, PKA, and TRNG.

AHB_Slave

AHB_Slave is used to configure registers. This module is in HCLK domain.

DMA

DMA is used to transfer data from external memory to RX_FIFO, or from TX_FIFO to external memory. DMA uses 64-bits AXI3 protocol with max burst length to 16. LLI transfer is also supported for performance and convenience consideration. This module is in ACLK domain.

CIPHER

CIPHER contains AES, DES/TDES and HASH engines. And it also supports various mode operations. The source data is either from RX_FIFO, or from other engine output. The result data is sending either to TX_FIFO, or Registers in module AHB_Slave. This module is

in CLK_CORE domain.

PKA

PKA is used to accelerate mathematical operations for big numbers. It supports - Modular arithmetic (addition, subtraction, multiplication and division), Regular arithmetic (addition, subtraction, multiplication and division), Modular inversion, Modular exponentiation, Logical operations (AND, OR, XOR, SHIFT). PKA has a SRAM which is used to store source, result and intermediate data for PKA operations. The software driver could use PKA operations to implement complicate calculation, such as RSA, ECC etc. It could support up to 4096 bits RSA modular exponentiation calculation. This module is in CLK_PKA domain.

TRNG

TRNG is used to collects random bits from the ring oscillator, up to 256 random bits per time. This module is in HCLK domain.

3.3 Register description

3.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_KEY_0	0x0180	W	0x00000000	Channel n range from 0 to 7. CHn_KEY_0 address = 0x0180 + 0x10 * n

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_KEY_1	0x0184	W	0x00000000	Channel n range from 0 to 7. CHn_KEY_1 address = 0x0184 + 0x10 * n.
CRYPTO_CHn_KEY_2	0x0188	W	0x00000000	Channel n range from 0 to 7. CHn_KEY_2 address = 0x0188 + 0x10 * n.

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_KEY_3	0x018c	W	0x00000000	Channel n range from 0 to 7.CHn_KEY_3 address = 0x018c + 0x10 * n.
CRYPTO_CHn_PC_LEN_0	0x0280	W	0x00000000	CHn_PC_LEN_0 address = 0x0280 + 0x8 * n, n = 0. Length in byte unit.

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_PC_LEN_1	0x0284	W	0x00000000	CHn_PC_LEN_1 address = 0x0284 + 0x8 * n, n = 0. Length in byte unit.
CRYPTO_CHn_ADA_LEN_0	0x02c0	W	0x00000000	CHn_ADA_LEN_0 address = 0x02c0 + 0x8 * n, n = 0. Length in byte unit.

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_ADA_LEN_1	0x02c4	W	0x00000000	CHn_ADA_LEN_1 address = 0x02c4 + 0x8 * n, n = 0. Length in byte unit.
CRYPTO_CHn_IV_LEN_0	0x0300	W	0x00000000	CHn_IV_LEN_0 address = 0x0300 + 0x4 * n, n = 0. Length in byte unit. Up to 16 byte IV for GCM.

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_TAG_0	0x0320	W	0x00000000	CHn_TAG_0 address = 0x0320 + 0x10 * n, n = 0. When the corresponding TAG_VALID is high, TAG value is valid.
CRYPTO_CHn_TAG_1	0x0324	W	0x00000000	CHn_TAG_1 address = 0x0324 + 0x10 * n, n = 0. When the corresponding TAG_VALID is high, TAG value is valid.

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_TAG_2	0x0328	W	0x00000000	CHn_TAG_2 address = 0x0328 + 0x10 * n, n = 0. When the corresponding TAG_VALID is high, TAG value is valid.
CRYPTO_CHn_TAG_3	0x032c	W	0x00000000	CHn_TAG_3 address = 0x032c + 0x10 * n, n = 0. When the corresponding TAG_VALID is high, TAG value is valid.

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_HASH_DOUT_0	0x03a0	W	0x00000000	HASH Data Output Register 0
CRYPTO_HASH_DOUT_1	0x03a4	W	0x00000000	HASH Data Output Register 1
CRYPTO_HASH_DOUT_2	0x03a8	W	0x00000000	HASH Data Output Register 2
CRYPTO_HASH_DOUT_3	0x03ac	W	0x00000000	HASH Data Output Register 3
CRYPTO_HASH_DOUT_4	0x03b0	W	0x00000000	HASH Data Output Register 4
CRYPTO_HASH_DOUT_5	0x03b4	W	0x00000000	HASH Data Output Register 5
CRYPTO_HASH_DOUT_6	0x03b8	W	0x00000000	HASH Data Output Register 6

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_HASH_DOUT_7	0x03bc	W	0x00000000	HASH Data Output Register 7
CRYPTO_HASH_DOUT_8	0x03c0	W	0x00000000	HASH Data Output Register 8
CRYPTO_HASH_DOUT_9	0x03c4	W	0x00000000	HASH Data Output Register 9
CRYPTO_HASH_DOUT_10	0x03c8	W	0x00000000	HASH Data Output Register 10
CRYPTO_HASH_DOUT_11	0x03cc	W	0x00000000	HASH Data Output Register 11
CRYPTO_HASH_DOUT_12	0x03d0	W	0x00000000	HASH Data Output Register 12
CRYPTO_HASH_DOUT_13	0x03d4	W	0x00000000	HASH Data Output Register 13

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_HASH_DOUT_14	0x03d8	W	0x00000000	HASH Data Output Register 14
CRYPTO_HASH_DOUT_15	0x03dc	W	0x00000000	HASH Data Output Register 15
CRYPTO_TAG_VALID	0x03e0	W	0x00000000	TAG Valid Register
CRYPTO_HASH_VALID	0x03e4	W	0x00000000	HASH Output Valid Register
CRYPTO_VERSION	0x03f0	W	0x01000001	CRYPTO Version Number Register
CRYPTO_RNG_CTL	0x0400	W	0x0000000c	RNG Control Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_RNG_SAMPLE_CNT	0x0404	W	0x00000000	RNG Sample Counter Register
CRYPTO_RNG_DOUT_0	0x0410	W	0x00000000	RNG Data Output Register 0
CRYPTO_RNG_DOUT_1	0x0414	W	0x00000000	RNG Data Output Register 1
CRYPTO_RNG_DOUT_2	0x0418	W	0x00000000	RNG Data Output Register 2
CRYPTO_RNG_DOUT_3	0x041c	W	0x00000000	RNG Data Output Register 3
CRYPTO_RNG_DOUT_4	0x0420	W	0x00000000	RNG Data Output Register 4

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_RNG_DOUT_5	0x0424	W	0x00000000	RNG Data Output Register 5
CRYPTO_RNG_DOUT_6	0x0428	W	0x00000000	RNG Data Output Register 6
CRYPTO_RNG_DOUT_7	0x042c	W	0x00000000	RNG Data Output Register 7
CRYPTO_RAM_CTL	0x0480	W	0x00000000	RAM Control Register
CRYPTO_RAM_ST	0x0484	W	0x00000001	RAM Status Register
CRYPTO_DEBUG_CTL	0x04a0	W	0x00000000	PKA Debug Control Register
CRYPTO_DEBUG_ST	0x04a4	W	0x00000001	PKA Debug Status Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_DEBUG_MONITOR	0x04a8	W	0x0000feef	PKA Debug Monitor Bus Register
CRYPTO_PKA_MEM_MAP0	0x0800	W	0x00000000	PKA Memory Map 0 Register
CRYPTO_PKA_MEM_MAP1	0x0804	W	0x00000000	PKA Memory Map 1 Register
CRYPTO_PKA_MEM_MAP2	0x0808	W	0x00000000	PKA Memory Map 2 Register
CRYPTO_PKA_MEM_MAP3	0x080c	W	0x00000000	PKA Memory Map 3 Register
CRYPTO_PKA_MEM_MAP4	0x0810	W	0x00000000	PKA Memory Map 4 Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_PKA_MEM_MAP5	0x0814	W	0x00000000	PKA Memory Map 5 Register
CRYPTO_PKA_MEM_MAP6	0x0818	W	0x00000000	PKA Memory Map 6 Register
CRYPTO_PKA_MEM_MAP7	0x081c	W	0x00000000	PKA Memory Map 7 Register
CRYPTO_PKA_MEM_MAP8	0x0820	W	0x00000000	PKA Memory Map 8 Register
CRYPTO_PKA_MEM_MAP9	0x0824	W	0x00000000	PKA Memory Map 9 Register
CRYPTO_PKA_MEM_MAP10	0x0828	W	0x00000000	PKA Memory Map 10 Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_PKA_MEM_MAP11	0x082c	W	0x00000000	PKA Memory Map 11 Register
CRYPTO_PKA_MEM_MAP12	0x0830	W	0x00000000	PKA Memory Map 12 Register
CRYPTO_PKA_MEM_MAP13	0x0834	W	0x00000000	PKA Memory Map 13 Register
CRYPTO_PKA_MEM_MAP14	0x0838	W	0x00000000	PKA Memory Map 14 Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_PKA_MEM_MAP15	0x083c	W	0x00000000	PKA Memory Map 15 Register
CRYPTO_PKA_MEM_MAP16	0x0840	W	0x00000000	PKA Memory Map 16 Register
CRYPTO_PKA_MEM_MAP17	0x0844	W	0x00000000	PKA Memory Map 17 Register
CRYPTO_PKA_MEM_MAP18	0x0848	W	0x00000000	PKA Memory Map 18 Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_PKA_MEM_MAP19	0x084c	W	0x00000000	PKA Memory Map 19 Register
CRYPTO_PKA_MEM_MAP20	0x0850	W	0x00000000	PKA Memory Map 20 Register
CRYPTO_PKA_MEM_MAP21	0x0854	W	0x00000000	PKA Memory Map 21 Register
CRYPTO_PKA_MEM_MAP22	0x0858	W	0x00000000	PKA Memory Map 22 Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_PKA_MEM_MAP23	0x085c	W	0x00000000	PKA Memory Map 23 Register
CRYPTO_PKA_MEM_MAP24	0x0860	W	0x00000000	PKA Memory Map 24 Register
CRYPTO_PKA_MEM_MAP25	0x0864	W	0x00000000	PKA Memory Map 25 Register
CRYPTO_PKA_MEM_MAP26	0x0868	W	0x00000000	PKA Memory Map 26 Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_PKA_MEM_MAP27	0x086c	W	0x00000000	PKA Memory Map 27 Register
CRYPTO_PKA_MEM_MAP28	0x0870	W	0x00000000	PKA Memory Map 28 Register
CRYPTO_PKA_MEM_MAP29	0x0874	W	0x00000000	PKA Memory Map 29 Register
CRYPTO_PKA_MEM_MAP30	0x0878	W	0x00000000	PKA Memory Map 30 Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_PKA_MEM_MAP31	0x087c	W	0x00000000	PKA Memory Map 31 Register
CRYPTO_PKA_OPCODE	0x0880	W	0x00000000	PKA Operation Code Register
CRYPTO_N_NP_TO_T1_ADDR	0x0884	W	0x000ff820	N_NP_TO_T1_ADDR Register
CRYPTO_PKA_STATUS	0x0888	W	0x00000001	PKA Status Register
CRYPTO_PKA_SW_RESET	0x088c	W	0x00000000	software reset of PKA
CRYPTO_PKA_L0	0x0890	W	0x00000000	PKA Length 0 Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_PKA_L1	0x0894	W	0x00000000	PKA Length 1 Register
CRYPTO_PKA_L2	0x0898	W	0x00000000	PKA Length 2 Register
CRYPTO_PKA_L3	0x089c	W	0x00000000	PKA Length 3 Register
CRYPTO_PKA_L4	0x08a0	W	0x00000000	PKA Length 4 Register
CRYPTO_PKA_L5	0x08a4	W	0x00000000	PKA Length 5 Register
CRYPTO_PKA_L6	0x08a8	W	0x00000000	PKA Length 6 Register
CRYPTO_PKA_L7	0x08ac	W	0x00000000	PKA Length 7 Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_PKA_PIPE_RDY	0x08b0	W	0x00000001	PKA pipe is ready for new opcode.
CRYPTO_PKA_DONE	0x08b4	W	0x00000001	PKA Done Register
CRYPTO_PKA_MON_SELECT	0x08b8	W	0x00000000	PKA Monitor Select Register
CRYPTO_PKA_DEBUG_REG_EN	0x08bc	W	0x00000000	PKA Debug Enable Register
CRYPTO_DEBUG_CNT_ADDR	0x08c0	W	0x00000000	Debug Counter Address Register

Name	Offset	Size	Reset Value	Description
CRYPTO_CLK_CTL	0x0000	W	0x00000001	Clock Control Register
CRYPTO_RST_CTL	0x0004	W	0x00000000	Reset Control Register
CRYPTO_DMA_INT_EN	0x0008	W	0x00000000	DMA Interrupt Enable Register
CRYPTO_DMA_INT_ST	0x000c	W	0x00000000	DMA Interrupt Status Register
CRYPTO_DMA_CTL	0x0010	W	0x00000000	DMA Control Register
CRYPTO_DMA_LLI_ADDR	0x0014	W	0x00000000	DMA LIST Start Address Register
CRYPTO_DMA_ST	0x0018	W	0x00000000	DMA Status Register
CRYPTO_DMA_STATE	0x001c	W	0x00000000	DMA State Register
CRYPTO_DMA_LLI_RADDR	0x0020	W	0x00000000	DMA LLI Read Address Register
CRYPTO_DMA_SRC_RADDR	0x0024	W	0x00000000	DMA Source Data Read Address Register
CRYPTO_DMA_DST_WADDR	0x0028	W	0x00000000	DMA Destination Data Read Address Register
CRYPTO_DMA_ITEM_ID	0x002c	W	0x00000000	DMA Descriptor ID Register
CRYPTO_FIFO_CTL	0x0040	W	0x00000003	FIFO Control Register
CRYPTO_BC_CTL	0x0044	W	0x00000000	Block Cipher Control Register
CRYPTO_HASH_CTL	0x0048	W	0x00000004	Hash Control Register
CRYPTO_CIPHER_ST	0x004c	W	0x00000000	Cipher Status Register
CRYPTO_CIPHER_STATE	0x0050	W	0x00000400	Cipher Current State Register
CRYPTO_CHn_IV_0	0x0100	W	0x00000000	Channel n IV Register 0, n = 0. CHn_IV_0 address = 0x0100 + 0x10 * n. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_1	0x0104	W	0x00000000	CHn_IV_1 address = 0x0104 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_2	0x0108	W	0x00000000	CHn_IV_2 address = 0x0108 + 0x10 * n, n = 0. For CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_CHn_IV_3	0x010c	W	0x00000000	CHn_IV_3 address = 0x010c + 0x10 * n, n = 0. or CTR Mode, IV stands for counter. For XTS Mode, IV stands for tweak.
CRYPTO_DEBUG_EXT_ADDR	0x08c4	W	0x00000000	Debug Extra Address Register
CRYPTO_PKA_DEBUG_HALT	0x08c8	W	0x00000000	PKA Debug Halt State Register
CRYPTO_PKA_MON_READ	0x08d0	W	0x0000feef	PKA Monitor Read Register
CRYPTO_PKA_INT_ENA	0x08d4	W	0x00000000	PKA Interrupt Enable Register
CRYPTO_PKA_INT_ST	0x08d8	W	0x00000000	PKA Interrupt Status Register
CRYPTO_SRAM_ADDR	0x1000	W	0x00000000	SRAM Base Address

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

3.3.2 Detail Register Description

CRYPTO_CLK_CTL

Address: Operational Base + offset (0x0000)

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved
16	WO	0x0	write_enable When bit n=1, the corresponding bit n-16 can be written by software; When bit n=0, the corresponding bit n-16 can't be written by software.
15:1	RO	0x0	reserved
0	RW	0x1	auto_clkgate_en CRYPTO will gate unused Block Cipher and HASH module automatically. 1'b0: disable; 1'b1: enable.

CRYPTO_RST_CTL

Address: Operational Base + offset (0x0004)

Bit	Attr	Reset Value	Description
31:19	RO	0x0	reserved
18:16	RW	0x0	write_enable When bit n=1, the corresponding bit n-16 can be written by software; When bit n=0, the corresponding bit n-16 can't be written by software.
15:3	RO	0x0	reserved
2	R/W SC	0x0	sw_pka_reset Software set this bit to start a reset to PKA module. After the reset is done, CRYPTO will clear this bit.
1	R/W SC	0x0	sw_rng_reset Software set this bit to start a reset to TRNG module. After the reset is done, CRYPTO will clear this bit.
0	R/W SC	0x0	sw_cc_reset Software set this bit to start a reset to Symmetric Cipher and HASH module. After the reset is done, CRYPTO will clear this bit.

CRYPTO_DMA_INT_EN

Address: Operational Base + offset (0x0008)

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	zero_len_int_en 1'b1: Enable; 1'b0: Disable.

Bit	Attr	Reset Value	Description
5	RW	0x0	list_err_int_en 1'b1: Enable; 1'b0: Disable.
4	RW	0x0	src_err_int_en 1'b1: Enable; 1'b0: Disable.
3	RW	0x0	dst_err_int_en 1'b1: Enable; 1'b0: Disable.
2	RW	0x0	src_item_done_int_en 1'b1: Enable; 1'b0: Disable.
1	RW	0x0	dst_item_done_int_en 1'b1: Enable; 1'b0: Disable.
0	RW	0x0	list_done_int_en 1'b1: Enable; 1'b0: Disable.

CRYPTO_DMA_INT_ST

Address: Operational Base + offset (0x000c)

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	W1C	0x0	zero_len 1'b1: Indicate that DMA has met an 0 byte source transfer length in list descriptors. After the bit is read, the application should write 1 to clear this bit for next time use. 1'b0: Nothing.
5	RO	0x0	reserved
4	W1C	0x0	src_err 1'b1: Indicate that DMA has met an error response when transfer source data. The state machine will exit current transfer and then return to IDLE state. After the bit is read, the application should write 1 to clear this bit for next time use. 1'b0: Nothing.
3	W1C	0x0	dst_err 1'b1: Indicate that DMA has met an error response when transfer destination data. The state machine will exit current transfer and then return to IDLE state. After the bit is read, the application should write 1 to clear this bit for next time use; 1'b0: Nothing.

Bit	Attr	Reset Value	Description
2	W1C	0x0	src_item_done 1'b1：Indicate that DMA has completed a read transfers which the current list descriptor pointed to. After the bit is read, the application should write 1 to clear this bit for next time use. 1'b0: Nothing.
1	W1C	0x0	dst_item_done 1'b1: Indicate that DMA has completed a write transfers which the current list descriptor pointed to. After the bit is read, the application should write 1 to clear this bit for next time use; 1'b0: Nothing.
0	W1C	0x0	list_done 1'b1: Indicate that DMA has completed all the transfers which the list descriptors pointed to. After the bit is read, the application should write 1 to clear this bit for next time use ; 1'b0: Nothing.

CRYPTO_DMA_CTL

Address: Operational Base + offset (0x0010)

Bit	Attr	Reset Value	Description
31:18	RO	0x0	reserved
17:16	WO	0x0	write_enable When bit n=1, the corresponding bit n-16 can be written by software; When bit n=0, the corresponding bit n-16 can't be written by software.
15:2	RO	0x0	reserved
1	R/WSC	0x0	dma_restart If DMA data for next stage is not ready, application could pause DMA by descriptor commands. DMA will stop prefetching next descriptor. The application could restart DMA by asserting this bit when DMA data for next state is ready. Crypto will continue with previous transfer, and clear the bit automatically.
0	R/WSC	0x0	dma_start DMA asserts the bit to start DMA transfer, then Crypto will clear the bit automatically.

CRYPTO_DMA_LLI_ADDR

Address: Operational Base + offset (0x0014)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	dma_lli_addr When DMA_CTL start asserted, Crypto will read the address to get the 1'st descriptor. It should be 8-bytes align. We suggest dma_lli_addr 64-byte align for best performance consideration.

CRYPTO_DMA_ST

Address: Operational Base + offset (0x0018)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	dma_busy 1'b1: Dma busy 1'b0: Dma idle

CRYPTO_DMA_STATE

Address: Operational Base + offset (0x001c)

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5:4	RO	0x0	dma_lli_state For debug use only. 2'b00: IDLE STATE; 2'b01: FETCH STATE; 2'b10: WORK STATE; Others: Reserved.
3:2	RO	0x0	dma_src_state For debug use only. 2'b00: IDLE STATE; 2'b01: LOAD STATE; 2'b10: WORK STATE; Others: Reserved.
1:0	RO	0x0	dma_dst_state For debug use only. 2'b00: IDLE STATE; 2'b01: LOAD STATE; 2'b10: WORK STATE; Others: Reserved.

CRYPTO_DMA_LLI_RADDR

Address: Operational Base + offset (0x0020)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	dma_lli_raddr For debug use only. It indicates the current dma lli read address.

CRYPTO_DMA_SRC_RADDR

Address: Operational Base + offset (0x0024)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	dma_src_raddr For debug use only. It indicates the current dma source read address.

CRYPTO_DMA_DST_WADDR

Address: Operational Base + offset (0x0028)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	dma_dst_waddr For debug use only. It indicates the current dma destination write address.

CRYPTO_DMA_ITEM_ID

Address: Operational Base + offset (0x002c)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0x00	dma_item_id For debug use only. It indicates the current descriptor ID.

CRYPTO_FIFO_CTL

Address: Operational Base + offset (0x0040)

Bit	Attr	Reset Value	Description
31:18	RO	0x0	reserved
17:16	WO	0x0	write_enable When bit n=1, the corresponding bit n-16 can be written by software; When bit n=0, the corresponding bit n-16 can't be written by software.
15:2	RO	0x0	reserved
1	RW	0x1	dout_byteswap 1'b1: Little endian; 1'b0: Big endian.
0	RW	0x1	din_byteswap 1'b1: Little endian; 1'b0: Big endian.

CRYPTO_BC_CTL

Address: Operational Base + offset (0x0044)

Bit	Attr	Reset Value	Description
31:26	RO	0x0	reserved
25:16	WO	0x000	write_enable When bit n=1, the corresponding bit n-16 can be written by software; When bit n=0, the corresponding bit n-16 can't be written by software.
15:10	RO	0x0	reserved

Bit	Attr	Reset Value	Description
9:8	RW	0x0	bc_cipher_sel 2'b00: AES; 2'b10: DES; 2'b11: TDES; Others:Reserved.
7:4	RW	0x0	mode For AES, 4'h0: ECB; 4'h1: CBC; 4'h2: CTS; 4'h3: CTR; 4'h4: CFB; 4'h5: OFB; 4'h6: XTS; 4'h7: CCM; 4'h8: GCM; 4'h9: CMAC; 4'hA: CBC-MAC; Others: Reserved. For TDES/DES, 4'h0: ECB; 4'h1: CBC; 4'h4: CFB; 4'h5: OFB; Others: Reserved.
3:2	RW	0x0	key_size For AES, 2'b00: 128 bit; 2'b01: 192 bit; 2'b10: 256 bit; 2'b11: Reserved; For TDES/DES, it is reserved.
1	RW	0x0	decrypt 1'b1: Decrypt; 1'b0: Encrypt.
0	RW	0x0	bc_enable 1'b1: Enable; 1'b0: Disable.

CRYPTO_HASH_CTL

Address: Operational Base + offset (0x0048)

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved

Bit	Attr	Reset Value	Description
23:16	WO	0x00	write_enable When bit n=1, the corresponding bit n-16 can be written by software; When bit n=0, the corresponding bit n-16 can't be written by software.
15:8	RO	0x0	reserved
7:4	RW	0x0	hash_cipher_sel 4'h0: SHA-1; 4'h1: MD-5; 4'h2: SHA-256; 4'h3: SHA-224; 4'h8: SHA-512; 4'h9: SHA-384; 4'hA: SHA-512/224; 4'hB: SHA-512/256; Others: Reserved.
3	RW	0x0	hmac_enable Crypto supports HMAC-SHA1, HMAC-SHA256, HMAC_SHA512. 1'b1: Enable; 1'b0: Disable.
2	RW	0x1	hw_pad_enable 1'b1: Enable; 1'b0: Disable.
1	RW	0x0	hash_src_sel 1'b1: From TX-FIFO; 1'b0: From RX-FIFO.
0	RW	0x0	hash_enable 1'b1: Enable; 1'b0: Disable.

CRYPTO_CIPHER_ST

Address: Operational Base + offset (0x004c)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RO	0x0	otp_key_valid Indicate if otp_key is valid. 1'b1: Valid 1'b0: Invalid
1	RO	0x0	hash_busy 1'b1: Busy 1'b0: Idle
0	RO	0x0	block_cipher_busy 1'b1: Busy 1'b0: Idle

CRYPTO_CIPHER_STATE

Address: Operational Base + offset (0x0050)

Bit	Attr	Reset Value	Description
31:15	RO	0x0	reserved
14:10	RO	0x01	hash_state For debug use only, 5'h01: IDLE State; 5'h02: IPAD State; 5'h04: TEXT State; 5'h08: OPAD State; 5'h10: OPAD EXT State.
9:8	RO	0x0	gcm_state For debug use only, 2'b00: IDLE State; 2'b01: PRE State; 2'b10: NA State; 2'b11: PC State.
7:6	RO	0x0	ccm_state For debug use only, 2'b00: IDLE State; 2'b01: PRE State; 2'b10: NA State; 2'b11: PC State.
5:4	RO	0x0	parallel_state For debug use only, 2'b00: IDLE State; 2'b01: PRE State; 2'b10: BULK State; Others: Reserved.
3:2	RO	0x0	mac_state For debug use only, 2'b00: IDLE State; 2'b01: PRE State; 2'b10: BULK State; Others: Reserved.
1:0	RO	0x0	serial_state For debug use only, 2'b00: IDLE State; 2'b01: PRE State; 2'b10: BULK State; 2'b11: Reserved.

CRYPTO_CHn_IV_0

Address: Operational Base + offset (0x0100)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	chn_iv_0 Channel N IV[127:96].

CRYPTO_CHn_IV_1

Address: Operational Base + offset (0x0104)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	chn_iv_1 Channel N IV[95:64].

CRYPTO_CHn_IV_2

Address: Operational Base + offset (0x0108)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	chn_iv_2 Channel N IV[63:32].

CRYPTO_CHn_IV_3

Address: Operational Base + offset (0x010c)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	chn_iv_3 Channel N IV[31:0].

CRYPTO_CHn_KEY_0

Address: Operational Base + offset (0x0180)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	chn_key_0 Channel N Key[127:96].

CRYPTO_CHn_KEY_1

Address: Operational Base + offset (0x0184)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	chn_key_1 Channel N Key[95:64].

CRYPTO_CHn_KEY_2

Address: Operational Base + offset (0x0188)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	chn_key_2 Channel N Key[63:32].

CRYPTO_CHn_KEY_3

Address: Operational Base + offset (0x018c)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	chn_key_3 Channel N Key[31:0].

CRYPTO_CHn_PC_LEN_0

Address: Operational Base + offset (0x0280)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	chn_pc_len_0 Channel N Plain/Cipher Text Length[31:0]

CRYPTO_CHn_PC_LEN_1

Address: Operational Base + offset (0x0284)

Bit	Attr	Reset Value	Description
31:29	RO	0x0	reserved
28:0	RW	0x00000000	chn_pc_len_1 Channel N Plain/Cipher Text Length[60:32].

CRYPTO_CHn_ADA_LEN_0

Address: Operational Base + offset (0x02c0)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	chn_ada_len_0 Channel N additional data Length[31:0].

CRYPTO_CHn_ADA_LEN_1

Address: Operational Base + offset (0x02c4)

Bit	Attr	Reset Value	Description
31:29	RO	0x0	reserved
28:0	RW	0x00000000	chn_ada_len_1 Channel N additional data Length[60:32].

CRYPTO_CHn_IV_LEN_0

Address: Operational Base + offset (0x0300)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:0	RW	0x00	chn_iv_len Channel N initial vector length.

CRYPTO_CHn_TAG_0

Address: Operational Base + offset (0x0320)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	chn_tag_0 Channel N tag[127:96].

CRYPTO_CHn_TAG_1

Address: Operational Base + offset (0x0324)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	chn_tag_1 Channel N tag[95:64].

CRYPTO_CHn_TAG_2

Address: Operational Base + offset (0x0328)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	chn_tag_2 Channel N tag[63:32].

CRYPTO_CHn_TAG_3

Address: Operational Base + offset (0x032c)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	chn_tag_3 Channel N tag[31:0].

CRYPTO_HASH_DOUT_0

Address: Operational Base + offset (0x03a0)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_0 0'th output word for all hash function, in big endian

CRYPTO_HASH_DOUT_1

Address: Operational Base + offset (0x03a4)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_1 1'th output word for all hash function, in big endian

CRYPTO_HASH_DOUT_2

Address: Operational Base + offset (0x03a8)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_2 2'th output word for all hash function, in big endian

CRYPTO_HASH_DOUT_3

Address: Operational Base + offset (0x03ac)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_3 3'th output word for all hash function, in big endian. This is MD5 last output word. HASH_DOUT_4 ~ HASH_DOUT_15 is invalid data for MD5.

CRYPTO_HASH_DOUT_4

Address: Operational Base + offset (0x03b0)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_4 4'th output word for all hash function, in big endian. This is SHA-1 last output word. HASH_DOUT_5 ~ HASH_DOUT_15 is invalid data for SHA-1.

CRYPTO_HASH_DOUT_5

Address: Operational Base + offset (0x03b4)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_5 5'th output word for all hash function, in big endian

CRYPTO_HASH_DOUT_6

Address: Operational Base + offset (0x03b8)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_6 6'th output word for all hash function, in big endian

CRYPTO_HASH_DOUT_7

Address: Operational Base + offset (0x03bc)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_7 7'th output word for all hash function, in big endian. This is SHA-256/224 last output word. HASH_DOUT_8 ~ HASH_DOUT_15 is invalid data for SHA-256/224.

CRYPTO_HASH_DOUT_8

Address: Operational Base + offset (0x03c0)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_8 8'th output word for all hash function, in big endian

CRYPTO_HASH_DOUT_9

Address: Operational Base + offset (0x03c4)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_9 9'th output word for all hash function, in big endian

CRYPTO_HASH_DOUT_10

Address: Operational Base + offset (0x03c8)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_10 10'th output word for all hash function, in big endian

CRYPTO_HASH_DOUT_11

Address: Operational Base + offset (0x03cc)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_11 11'th output word for all hash function, in big endian.

CRYPTO_HASH_DOUT_12

Address: Operational Base + offset (0x03d0)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_12 12'th output word for all hash function, in big endian

CRYPTO_HASH_DOUT_13

Address: Operational Base + offset (0x03d4)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_13 13'th output word for all hash function, in big endian

CRYPTO_HASH_DOUT_14

Address: Operational Base + offset (0x03d8)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_14 14'th output word for all hash function, in big endian

CRYPTO_HASH_DOUT_15

Address: Operational Base + offset (0x03dc)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	hash_dout_15 15'th output word for all hash function, in big endian. This is SHA-512, SHA-384, SHA-512/224, SHA-512/256 last output word.

CRYPTO_TAG_VALID

Address: Operational Base + offset (0x03e0)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	W1C	0x0	ch7_tag_valid When channel 7 tag calculation is finished, CRYPTO will set this bit. After the bit is read by application, it should be cleared by writing 1 to this bit. 1'b1: Channel 7 tag is valid 1'b0: Channel 7 tag is invalid
6	W1C	0x0	ch6_tag_valid When channel 6 tag calculation is finished, CRYPTO will set this bit. After the bit is read by application, it should be cleared by writing 1 to this bit. 1'b1: Channel 6 tag is valid 1'b0: Channel 6 tag is invalid

Bit	Attr	Reset Value	Description
5	W1C	0x0	ch5_tag_valid When channel 5 tag calculation is finished, CRYPTO will set this bit. After the bit is read by application, it should be cleared by writing 1 to this bit. 1'b1: Channel 5 tag is valid 1'b0: Channel 5 tag is invalid
4	W1C	0x0	ch4_tag_valid When channel 4 tag calculation is finished, CRYPTO will set this bit. After the bit is read by application, it should be cleared by writing 1 to this bit. 1'b1: Channel 4 tag is valid 1'b0: Channel 4 tag is invalid
3	W1C	0x0	ch3_tag_valid When channel 3 tag calculation is finished, CRYPTO will set this bit. After the bit is read by application, it should be cleared by writing 1 to this bit. 1'b1: Channel 3 tag is valid 1'b0: Channel 3 tag is invalid
2	W1C	0x0	ch2_tag_valid When channel 2 tag calculation is finished, CRYPTO will set this bit. After the bit is read by application, it should be cleared by writing 1 to this bit. 1'b1: Channel 2 tag is valid 1'b0: Channel 2 tag is invalid
1	W1C	0x0	ch1_tag_valid When channel 1 tag calculation is finished, CRYPTO will set this bit. After the bit is read by application, it should be cleared by writing 1 to this bit. 1'b1: Channel 1 tag is valid 1'b0: Channel 1 tag is invalid
0	W1C	0x0	ch0_tag_valid When channel 0 tag calculation is finished, CRYPTO will set this bit. After the bit is read by application, it should be cleared by writing 1 to this bit. 1'b1: Channel 0 tag is valid 1'b0: Channel 0 tag is invalid

CRYPTO_HASH_VALID

Address: Operational Base + offset (0x03e4)

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

Bit	Attr	Reset Value	Description
0	W1C	0x0	hash_valid When HASH calculation is finished, CRYPTO will set this bit. After the bit is read by application, it should be cleared by writing 1 to this bit. 1'b1: HASH_DOUT is valid 1'b0: HASH_DOUT is invalid

CRYPTO_VERSION

Address: Operational Base + offset (0x03f0)

Bit	Attr	Reset Value	Description
31:0	RW	0x01000001	version_num Version number: V1.0.0.1.

CRYPTO_RNG_CTL

Address: Operational Base + offset (0x0400)

Bit	Attr	Reset Value	Description
31:22	RO	0x0	reserved
21:16	WO	0x00	write_enable When bit n=1, the corresponding bit n-16 can be written by software; When bit n=0, the corresponding bit n-16 can't be written by software.
15:6	RO	0x0	reserved
5:4	RW	0x0	rng_len 2'b00: 64 bit 2'b01: 128 bit 2'b10: 192 bit 2'b11: 256 bit
3:2	RW	0x3	ring_sel There are 4 osc rings choice to decide the rng output data. 2'b00: Fatest osc ring 2'b01: Slower than osc ring 0 2'b10: Slower than osc ring 1 2'b11: Slowest osc ring
1	RW	0x0	rng_enable 1'b1: Enable 1'b0: Disable
0	R/WSC	0x0	rng_start The application triggers this bit to start collect rng output data. After rng is started, CRYPTO will clear the bit automatically. 1'b1: Start 1'b0: Do nothing

CRYPTO_RNG_SAMPLE_CNT

Address: Operational Base + offset (0x0404)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RW	0x0000	rng_sample_cnt RNG collects osc ring output bit every rng_sample_cnt time. The value of rng_sample_cnt affects RNG output data rate, the value more bigger, the rate more slower.

CRYPTO_RNG_DOUT_0

Address: Operational Base + offset (0x0410)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rng_dout_0 The 32'th osc ring bit is captured in RNG_DOUT_0.bit31.

CRYPTO_RNG_DOUT_1

Address: Operational Base + offset (0x0414)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rng_dout_1 The 64'th osc ring bit is captured in RNG_DOUT_1.bit31. If RNG_CTL.rng_len = 0x00, the last valid bit of RNG is stored in RNG_DOUT_1.bit31, and RNG_DOUT_2 ~ RNG_DOUT_7 are invalid.

CRYPTO_RNG_DOUT_2

Address: Operational Base + offset (0x0418)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rng_dout_2 The 96'th osc ring bit is captured in RNG_DOUT_2.bit31.

CRYPTO_RNG_DOUT_3

Address: Operational Base + offset (0x041c)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rng_dout_3 The 128'th osc ring bit is captured in RNG_DOUT_3.bit31. If RNG_CTL rng_len = 0x01, the last valid bit of RNG is stored in RNG_DOUT_3.bit31, and RNG_DOUT_4 ~ RNG_DOUT_7 are invalid.

CRYPTO_RNG_DOUT_4

Address: Operational Base + offset (0x0420)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rng_dout_4 The 160'th osc ring bit is captured in RNG_DOUT_4.bit31.

CRYPTO_RNG_DOUT_5

Address: Operational Base + offset (0x0424)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rng_dout_5 The 192'th osc ring bit is captured in RNG_DOUT_5.bit31. If RNG_CTL.rng_len = 0x02, the last valid bit of RNG is stored in RNG_DOUT_5.bit31, and RNG_DOUT_6~ RNG_DOUT_7 are invalid.

CRYPTO_RNG_DOUT_6

Address: Operational Base + offset (0x0428)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	rng_dout_6 The 224'th osc ring bit is captured in RNG_DOUT_6.bit31.

CRYPTO_RNG_DOUT_7

Address: Operational Base + offset (0x042c)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	rng_dout_7 The 256'th osc ring bit is captured in RNG_DOUT_7.bit31. If RNG_CTL.rng_len = 0x03, the last valid bit of RNG is stored in RNG_DOUT_7.bit31.

CRYPTO_RAM_CTL

Address: Operational Base + offset (0x0480)

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved
16	WO	0x0	write_enable When bit n=1, the corresponding bit n-16 can be written by software; When bit n=0, the corresponding bit n-16 can't be written by software.
15:1	RO	0x0	reserved
0	RW	0x0	ram_pka_rdy Indicate whether ram is controlled by PKA engine. 1'b0: Ram is controlled by CPU 1'b1: Ram is controlled by CRYPTO PKA engine

CRYPTO_RAM_ST

Address: Operational Base + offset (0x0484)

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

Bit	Attr	Reset Value	Description
0	RO	0x1	clk_ram_rdy Indicate whether clk_ram is stable, and ready for use. 1'b0: Not stable 1'b1: Stable

CRYPTO_DEBUG_CTL

Address: Operational Base + offset (0x04a0)

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved
16	WO	0x0	write_enable When bit n=1, the corresponding bit n-16 can be written by software; When bit n=0, the corresponding bit n-16 can't be written by software.
15:1	RO	0x0	reserved
0	RW	0x0	pka_debug_mode 1'b1: PKA is in debug mode 1'b0: PKA is in normal mode

CRYPTO_DEBUG_ST

Address: Operational Base + offset (0x04a4)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x1	pka_debug_clk_en For debug use only, 1'b1: Enable 1'b0: Disable

CRYPTO_DEBUG_MONITOR

Address: Operational Base + offset (0x04a8)

Bit	Attr	Reset Value	Description
31:0	RW	0x0000feef	pka_monitor_bus For debug use only.

CRYPTO_PKA_MEM_MAP0

Address: Operational Base + offset (0x0800)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map0 Memory map 0 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP1

Address: Operational Base + offset (0x0804)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map1 Memory map 1 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP2

Address: Operational Base + offset (0x0808)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map2 Memory map 2 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP3

Address: Operational Base + offset (0x080c)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map3 Memory map 3 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP4

Address: Operational Base + offset (0x0810)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map4 Memory map 4 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP5

Address: Operational Base + offset (0x0814)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map5 Memory map 5 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP6

Address: Operational Base + offset (0x0818)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map6 Memory map 6 [11:2], bit[1:0] is stuck to 0.

Bit	Attr	Reset Value	Description
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP7

Address: Operational Base + offset (0x081c)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map7 Memory map 7 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP8

Address: Operational Base + offset (0x0820)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map8 Memory map 8 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP9

Address: Operational Base + offset (0x0824)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map9 Memory map 9 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP10

Address: Operational Base + offset (0x0828)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map10 Memory map 10 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP11

Address: Operational Base + offset (0x082c)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map11 Memory map 11 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP12

Address: Operational Base + offset (0x0830)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map12 Memory map 12 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP13

Address: Operational Base + offset (0x0834)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map13 Memory map 13 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP14

Address: Operational Base + offset (0x0838)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map14 Memory map 14 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP15

Address: Operational Base + offset (0x083c)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map15 Memory map 15 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP16

Address: Operational Base + offset (0x0840)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map16 Memory map 16 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP17

Address: Operational Base + offset (0x0844)

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

Bit	Attr	Reset Value	Description
11:2	RW	0x000	memory_map17 Memory map 17 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP18

Address: Operational Base + offset (0x0848)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map18 Memory map 18 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP19

Address: Operational Base + offset (0x084c)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map19 Memory map 19 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP20

Address: Operational Base + offset (0x0850)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map20 Memory map 20 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP21

Address: Operational Base + offset (0x0854)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map21 Memory map 21 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP22

Address: Operational Base + offset (0x0858)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map22 Memory map 22 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP23

Address: Operational Base + offset (0x085c)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map23 Memory map 23 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP24

Address: Operational Base + offset (0x0860)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map24 Memory map 24 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP25

Address: Operational Base + offset (0x0864)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map25 Memory map 25 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP26

Address: Operational Base + offset (0x0868)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map26 Memory map 26 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP27

Address: Operational Base + offset (0x086c)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map27 Memory map 27 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP28

Address: Operational Base + offset (0x0870)

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

Bit	Attr	Reset Value	Description
11:2	RW	0x000	memory_map28 Memory map 28 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP29

Address: Operational Base + offset (0x0874)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map29 Memory map 29 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP30

Address: Operational Base + offset (0x0878)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map30 Memory map 30 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_MEM_MAP31

Address: Operational Base + offset (0x087c)

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11:2	RW	0x000	memory_map31 Memory map 30 [11:2], bit[1:0] is stuck to 0.
1:0	RO	0x0	reserved

CRYPTO_PKA_OPCODE

Address: Operational Base + offset (0x0880)

Bit	Attr	Reset Value	Description
31:27	WO	0x00	opcode Defines the PKA operation 5'h04: Add,Inc 5'h05: Sub,Dec,Neg 5'h06: ModAdd,ModInc 5'h07: ModSub,ModDec,ModNeg 5'h08: AND,TST0,CLR0 5'h09: OR,COPY,SET0 5'h0A: XOR,FLIP0,INVERT,COMPARE 5'h0B: SHR0 5'h0D: SHR1 5'h0E: SHL0 5'h0F: SHL1 5'h10: MulLow 5'h11: ModMul 5'h12: ModMulN 5'h13: ModExp 5'h14: Division 5'h15: Div 5'h16: ModDiv 5'h00: Terminate
26:24	WO	0x0	len The virtual length address 0-7. Virtual address 0 point to PKA_L0. Virtual address 1 point to PKA_L1. ... Virtual address 7 point to PKA_L7.
23:18	WO	0x00	reg_a Operand A virtual address 0-15. Virtual address 0 point to PKA_MEM_MAP0. Virtual address 1 point to PKA_MEM_MAP1. ... Virtual address 15 point to PKA_MEM_MAP15.
17:12	WO	0x00	reg_b Operand B virtual address 0-15. Virtual address 0 point to PKA_MEM_MAP0. Virtual address 1 point to PKA_MEM_MAP1. ... Virtual address 15 point to PKA_MEM_MAP15.
11:6	WO	0x00	reg_r Result register virtual address 0-15. Virtual address 0 point to PKA_MEM_MAP0. Virtual address 1 point to PKA_MEM_MAP1. ... Virtual address 15 point to PKA_MEM_MAP15.

Bit	Attr	Reset Value	Description
5:0	WO	0x00	tag Tag.

CRYPTO_N_NP_TO_T1_ADDR

Address: Operational Base + offset (0x0884)

Bit	Attr	Reset Value	Description
31:20	RO	0x0	reserved
19:15	RW	0x1f	reg_t1 Virtual address of temporary register number 1. Virtual address 0 point to PKA_MEM_MAP0. Virtual address 1 point to PKA_MEM_MAP1. ... Virtual address 15 point to PKA_MEM_MAP15.
14:10	RW	0x1e	reg_t0 Virtual address of temporary register number 0. Virtual address 0 point to PKA_MEM_MAP0. Virtual address 1 point to PKA_MEM_MAP1. ... Virtual address 15 point to PKA_MEM_MAP15.
9:5	RW	0x01	reg_np Virtual address of register np. Virtual address 0 point to PKA_MEM_MAP0. Virtual address 1 point to PKA_MEM_MAP1. ... Virtual address 15 point to PKA_MEM_MAP15.
4:0	RW	0x00	reg_n Virtual address of register n. Virtual address 0 point to PKA_MEM_MAP0. Virtual address 1 point to PKA_MEM_MAP1. ... Virtual address 15 point to PKA_MEM_MAP15.

CRYPTO_PKA_STATUS

Address: Operational Base + offset (0x0888)

Bit	Attr	Reset Value	Description
31:20	RO	0x0	reserved
19:14	RO	0x00	tag Tag of the Last Operation
13:9	RO	0x00	opcode The last OPCODE
8	RO	0x0	pka_cpu_busy PKA is busy memory control is by PKA.
7	RO	0x0	modinv_of_zero Modular inverse of zero flag

Bit	Attr	Reset Value	Description
6	RO	0x0	alu_sign_out Sign of the last operation(MSB)
5	RO	0x0	alu_carry Carry of the last ALU operation
4	RO	0x0	div_by_zero Division by 0
3	RO	0x0	alu_mod_ovflw Modular overflow flag
2	RO	0x0	alu_out_zero ALU out is 0.
1	RO	0x0	pka_busy PKA is busy.
0	RO	0x1	pipe_is_busy PKA ready signal 1'b0: Pipe full 1'b1: PKA ready for new command

CRYPTO_PKA_SW_RESET

Address: Operational Base + offset (0x088c)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	WO	0x0	pka_sw_reset PKA software reset the reset mechanism will take about four PKA clocks until the reset line is de-asserted.

CRYPTO_PKA_L0

Address: Operational Base + offset (0x0890)

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12:0	RW	0x0000	pka_l0 PKA length 0, in bit unit

CRYPTO_PKA_L1

Address: Operational Base + offset (0x0894)

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12:0	RW	0x0000	pka_l1 PKA length 1, in bit unit

CRYPTO_PKA_L2

Address: Operational Base + offset (0x0898)

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved

Bit	Attr	Reset Value	Description
12:0	RW	0x0000	pka_l2 PKA length 2, in bit unit

CRYPTO_PKA_L3

Address: Operational Base + offset (0x089c)

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12:0	RW	0x0000	pka_l3 PKA length 3, in bit unit

CRYPTO_PKA_L4

Address: Operational Base + offset (0x08a0)

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12:0	RW	0x0000	pka_l4 PKA length 4, in bit unit

CRYPTO_PKA_L5

Address: Operational Base + offset (0x08a4)

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12:0	RW	0x0000	pka_l5 PKA length 5, in bit unit

CRYPTO_PKA_L6

Address: Operational Base + offset (0x08a8)

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12:0	RW	0x0000	pka_l6 PKA length 6, in bit unit

CRYPTO_PKA_L7

Address: Operational Base + offset (0x08ac)

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12:0	RW	0x0000	pka_l7 PKA length 7, in bit unit

CRYPTO_PKA_PIPE_RDY

Address: Operational Base + offset (0x08b0)

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

Bit	Attr	Reset Value	Description
0	RO	0x1	pka_pipe_rdy PKA pipe is ready for new opcode.

CRYPTO_PKA_DONE

Address: Operational Base + offset (0x08b4)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x1	pka_done PKA operation is completed and pipe is empty.

CRYPTO_PKA_MON_SELECT

Address: Operational Base + offset (0x08b8)

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3:0	RW	0x0	pka_mon_select PKA monitor select which PKA fsm monitor is being output.

CRYPTO_PKA_DEBUG_REG_EN

Address: Operational Base + offset (0x08bc)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	pka_debug_reg_en Enable all the debug mechanism when set.

CRYPTO_DEBUG_CNT_ADDR

Address: Operational Base + offset (0x08c0)

Bit	Attr	Reset Value	Description
31:20	RO	0x0	reserved
19:0	R/WSC	0x00000	debug_cnt_addr The clock counter initial values. clock is disabled when counter expires. Triggered when pka_debug_en is set.

CRYPTO_DEBUG_EXT_ADDR

Address: Operational Base + offset (0x08c4)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	WO	0x0	debug_ext_addr Disable the debug Mechanism

CRYPTO_PKA_DEBUG_HALT

Address: Operational Base + offset (0x08c8)

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

Bit	Attr	Reset Value	Description
0	RO	0x0	pka_debug_halt In debug mode: PKA is in halt state.

CRYPTO_PKA_MON_READ

Address: Operational Base + offset (0x08d0)

Bit	Attr	Reset Value	Description
31:0	RO	0x0000feef	pka_mon_read This is the PKA monitor bus register output.

CRYPTO_PKA_INT_ENA

Address: Operational Base + offset (0x08d4)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	pka_int_ena 1'b1: Enable pka interrupt 1'b0: Disable pka interrupt

CRYPTO_PKA_INT_ST

Address: Operational Base + offset (0x08d8)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	W1C	0x0	pka_int_st Indicate that PKA operation completes. After the bit is read, the application should write 1 to clear this bit for next time use.

CRYPTO_SRAM_ADDR

Address: Operational Base + offset (0x1000)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	sram_addr Sram address starts from 0x1000 to 0x1fff. When RAM_CTL.ram_pka_rdy == 0, application could access sram. Otherwise, application can't.

3.4 Application Note

3.4.1 Clock &Reset

There are 4 clock domains in Crypto. The clock and reset signals are described in the following table.

Table 3-1 Crypto Clock & Reset Description

Signal	Attr	Description
hclk	clock	AHB clock
aclk	clock	AXI master clock

Signal	Attr	Description
clk_core	clock	Cipher work clock
clk_pka	clock	PKA work clock
hresetn	reset	Asynchronously assert, synchronously de-assert to hclk,low active
aresetn	reset	Asynchronously assert, synchronously de-assert to aclk,low active
resetn_core	reset	Asynchronously assert, synchronously de-assert to clk_core,low active
resetn_pka	reset	Asynchronously assert, synchronously de-assert to clk_pka,low active

Each function need different clocks. The applications could gate the un-used clock to save power. Please see the following table for detail information.

Table 3-2 Crypto Clock & Reset Description

Operation	HCLK	ACLK	CLK_CORE	CLK_PKA
AES	ON	ON	ON	OFF
DES/TDES	ON	ON	ON	OFF
HASH/HMAC	ON	ON	ON	OFF
PKA	ON	OFF	OFF	ON
TRNG	ON	OFF	OFF	OFF

Even when CLK_CORE is on, Crypto is doing some cipher job. And Crypto could still be able to automatically gate most parts of un-used blocks to save more power, if CRYPTO_CLK_CTL.auto_clkgate_en is set to '1'. The default value for this bit is also '1'. Application could do a soft reset to a certain clock domain. Please refer to "Chapter CRU" for more details.

3.4.2 Performance

Cipher performance is shown in the following table.

Table 3-3 Crypto Performance Description

Algorithm	block size (Byte)	clk_core frequency (Mhz)	cycle	serial max throughput (MBps)	parallel max throughput (MBps)
DES	8	200	18	88	352
TDES	8	200	55	29	116
AES-128	16	200	12	266	1066
AES-192	16	200	14	228	914
AES-256	16	200	16	200	800
SHA-1	64	200	81	158	NA
MD5	64	200	65	196	NA
SHA-256/224	64	200	65	196	NA
SHA-512/384/ 512_224/ 512_256	128	200	81	316	NA

There are 2 column throughput rates in the table, 1 is serial mode, and the other is parallel mode. In parallel mode, there are 4 engines working at the same time. So the speed is 4 times than serial mode. Parallel mode includes ECB/CTR/XTS both encryption and decryption mode, CFB/CBC/CTS only decryption mode. Other modes are serial. HASH doesn't have parallel mode.

For PKA, the cycles for each calculation are not certain. It depends on the parameters. Take RSA-2048 for example, it takes about 28M cycles to finish a calculation. PKA can run 300 MHz. It means it can run over 10 times per second.

3.4.3 DMA

DMA supports Link List Item (LLI) DMA transaction.

- Each item contains 8 bytes, and start address should be 8 bytes align;
- We suggest that DATA start address is 8 bytes align;
- Total DATA length is byte align;
- Support segmenting HASH/HMAC DATA into multi sections. We suggest that each section DATA length is a multiple of 64 bytes, except this section is the last section.

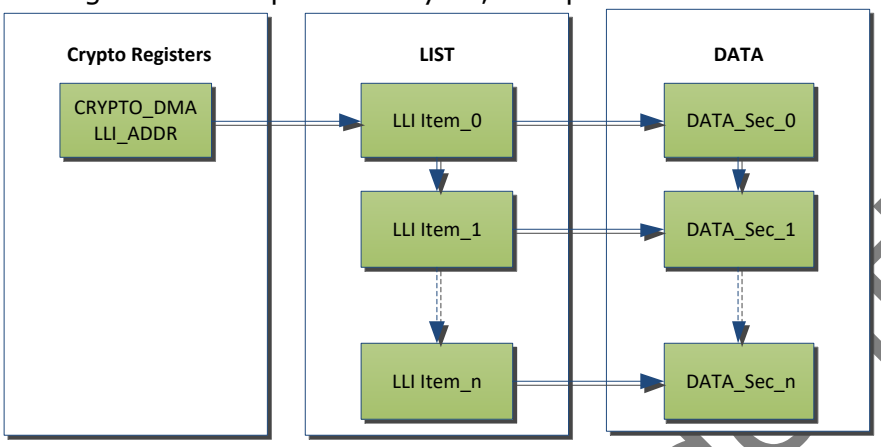

Fig.3-2 LLI DMA Usage

As shown in the Figure above, Register CRYPTO_DMA_LLI_ADDR points to 1'st LLI item in external memory. Each LLI item contains DATA address, length, control information and next LLI item pointer, except the last LLI item. The last item doesn't have the next LLI Item pointer. After the last LLI item is finished, DMA will go to idle state.
LLI item definition is shown in the following table.

Table 3-4 LLI Item Description

offset	Def	Description
0x00	SRC_ADDRESS[31:0]	Source data start address
0x04	SRC_LENGTH[31:0]	Source data length，in byte unit
0x08	DST_ADDRESS[31:0]	Destination data start address
0x0c	DST_LENGTH[31:0]	Destination data length，in byte unit
0x10	USER_DEFINE[31:0]	Used in cipher block
0x14	reserve	Reserve
0x18	DMA_CTRL[31:0]	Used in DMA block
0x1c	next address[31:0]	Next LLI item address. When DMA_CTRL.LAST = '1'，NEXT_ADDRESS is invalid.

DMA_CTRL: the definition is shown in the following table.

Table 3-5 LLI Item dma_ctl Description

Bit	Def	Definition
[31:24]	ITEM_ID[7:0]	Used to identify LLI items.
[23:16]	reserve	
[15:11]	reserve	
10	source_item_done enable	When source data fetch is completed, CRYPTO_DMA_INT_ST.source_item_done will assert if this bit is set.
9	reserved	reserved
8	list_done enable	When all LLI items transfer is completed, CRYPTO_DMA_INT_ST.list_done will assert if this bit is set.

Bit	Def	Definition
[7:2]	reserved	reserved
1	PAUSE	Indicate DMA will hold on after executes current item. DMA won't go on unless CRYPTO_DMA_CTL.restart is configured.
0	LAST	Indicate current item is the last one. After executes current item, DMA will return to IDLE state.

Table 3-6 LLI Item user_define Description

Bit	Signal	Description
31:9	Reserved	Reserved
8	otpkey_sel	Otpkey select. 1: select otpkey; 0: select register key
7	Privacy_sel	Pkey select. 1: select pkey; 0: select key;
6:4	Chnl_num	Channel number, from 0 to 7.
3	String_attr	Indicate current item's attribution. 0: ADA; 1: PC (plaintext or ciphertext).
2	String_last	Indicate current item is the string last item
1	String_start	Indicate current item is the string first item
0	Cipher_start	Indicate current item is the cipher first item

3.4.4 Multi-Channel Map

There are 8-channel configurations for AES or DES/TDES operation. For different key-size, the map is different. Please find the register map in the following table.

Table 3-7 LLI Item user_define Description

Cipher sel	otpkey sel	privacy sel	chnl num	key	iv(tag/…)
AES-128/ DES	0	0	0	CH0_KEY0-3/ CH0_KEY0-1	CH0_IV0-3/ CH0_IV0-1
AES-128/ DES	0	0	n	CHn_KEY0-3/ CHn_KEY0-1	CHn_IV0-3/ CHn_IV0-1
AES-128/ DES	0	0	7	CH7_KEY0-3/ CH7_KEY0-1	CH7_IV0-3/ CH7_IV0-1
AES-128/ DES	0	1	0	CH0_PKEY0-3/ CH0_PKEY0-1	CH0_IV0-3/ CH0_IV0-1
AES-128/ DES	0	1	n	CHn_PKEY0-3/ CHn_PKEY0-1	CHn_IV0-3/ CHn_IV0-1
AES-128/ DES	0	1	7	CH7_PKEY0-3/ CH7_PKEY0-1	CH7_IV0-3/ CH7_IV0-1
AES-128/ DES	1	NA	0	OTP_KEY[255:128]	CH0_IV0-3/ CH0_IV0-1
AES-128/ DES	1	NA	n	OTP_KEY[127:0]	CHn_IV0-3/ CHn_IV0-1
AES-128/ DES	1	NA	7	OTP_KEY[127:0]	CH7_IV0-3/ CH7_IV0-1
AES-192/ TDES	0	0	0	CH0_KEY0-3, CH1_KEY0-1	CH0_IV0-3/ CH0_IV0-1
AES-192/ TDES	0	0	1	CH2_KEY0-3, CH3_KEY0-1	CH1_IV0-3/ CH1_IV0-1
AES-192/ TDES	0	0	2	CH4_KEY0-3, CH5_KEY0-1	CH2_IV0-3/ CH2_IV0-1
AES-192/ TDES	0	0	3	CH6_KEY0-3, CH7_KEY0-1	CH3_IV0-3/ CH3_IV0-1

Cipher sel	otpkey sel	privacy sel	chnl num	key	iv(tag/…)
AES-192/ TDES	0	1	0	CH0_PKEY0-3, CH1_PKEY0-1	CH0_IV0-3/ CH0_IV0-1
AES-192/ TDES	0	1	1	CH2_PKEY0-3, CH3_PKEY0-1	CH1_IV0-3/ CH1_IV0-1
AES-192/ TDES	0	1	2	CH4_PKEY0-3, CH5_PKEY0-1	CH2_IV0-3/ CH2_IV0-1
AES-192/ TDES	0	1	3	CH6_PKEY0-3, CH7_PKEY0-1	CH3_IV0-3/ CH3_IV0-1
AES-192/ TDES	0	NA	4-7	not supported	not supported
AES-192/ TDES	1	NA	0	OTP[255:64]	CH0_IV0-3/ CH0_IV0-1
AES-192/ TDES	1	NA	1-7	not supported	not supported
AES-256	0	0	0	CH0_KEY0-3, CH1_KEY0-3	CH0_IV0-3/ CH0_IV0-1
AES-256	0	0	1	CH2_KEY0-3, CH3_KEY0-3	CH1_IV0-3/ CH1_IV0-1
AES-256	0	0	2	CH4_KEY0-3, CH5_KEY0-3	CH2_IV0-3/ CH2_IV0-1
AES-256	0	0	3	CH6_KEY0-3, CH7_KEY0-3	CH3_IV0-3/ CH3_IV0-1
AES-256	0	1	0	CH0_PKEY0-3, CH1_PKEY0-3	CH0_IV0-3/ CH0_IV0-1
AES-256	0	1	1	CH2_PKEY0-3, CH3_PKEY0-3	CH1_IV0-3/ CH1_IV0-1
AES-256	0	1	2	CH4_PKEY0-3, CH5_PKEY0-3	CH2_IV0-3/ CH2_IV0-1
AES-256	0	1	3	CH6_PKEY0-3, CH7_PKEY0-3	CH3_IV0-3/ CH3_IV0-1
AES-256	0	NA	4-7	not supported	not supported
AES-256	1	NA	0	OTP[255:0]	CH0_IV0-3/ CH0_IV0-1
AES-256	1	NA	1-7	not supported	not supported

In AES-XTS mode, there are 2 keys, and only AES-128 and AES-256 mode are. Please refer to the following table for detail information.

Table 3-8 LLI Item user_define Description

Cipher sel	otpkey sel	privacy sel	chnl num	key1	key2	tweak
AES-128	0	0	0	CH0_KEY0-3	CH4_KEY0-3	CH0_IV0-3
AES-128	0	0	1	CH1_KEY0-3	CH5_KEY0-3	CH1_IV0-3
AES-128	0	0	2	CH2_KEY0-3	CH6_KEY0-3	CH2_IV0-3
AES-128	0	0	3	CH3_KEY0-3	CH7_KEY0-3	CH3_IV0-3
AES-128	0	1	0	CH0_PKEY0-3	CH4_PKEY0-3	CH0_IV0-3
AES-128	0	1	1	CH1_PKEY0-3	CH5_PKEY0-3	CH1_IV0-3
AES-128	0	1	2	CH2_PKEY0-3	CH6_PKEY0-3	CH2_IV0-3

Cipher sel	otpkey sel	privacy sel	chnl num	key1	key2	tweak
AES-128	0	1	3	CH3_PKEY0-3	CH7_PKEY0-3	CH3_IV0-3
AES-128	0	NA	4-7	not supported	not supported	not supported
AES-128	1	NA	NA	not supported	not supported	not supported
AES-256	0	0	0	CH0_KEY0-3, CH1_KEY0-3	CH4_KEY0-CH5_KEY3	CH0_IV0-3
AES-256	0	0	1	CH2_KEY0-3, CH3_KEY0-3	CH6_KEY0-CH7_KEY3	CH1_IV0-3
AES-256	0	1	0	CH0_PKEY0-3, CH1_PKEY0-3	CH4_PKEY0-CH5_PKEY3	CH0_IV0-3
AES-256	0	1	1	CH2_PKEY0-3, CH3_PKEY0-3	CH6_PKEY0-CH7_PKEY3	CH1_IV0-3
AES-256	0	NA	2-7	not supported	not supported	not supported
AES-256	1	NA	NA	not supported	not supported	not supported

Note: The difference between CHn_KEY and CHn_PKEY is that: CHn_KEY could be read/write, CHn_PKEY could be write, but can't be read. The read value for CHn_PKEY is all '0'.

3.4.5 HASH Data Path

HASH and AES could run in parallel way. There are 2 paths lead to AES-HASH function. One is AES-HASH-RX mode, the other is AES-HASH-TX mode.

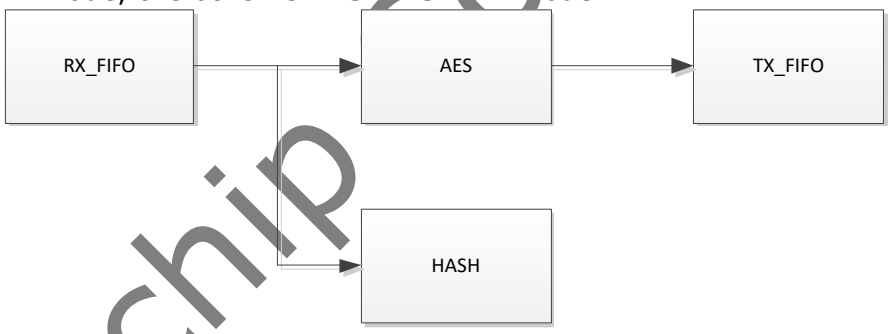

Fig.3-3 AES-HASH-RX mode

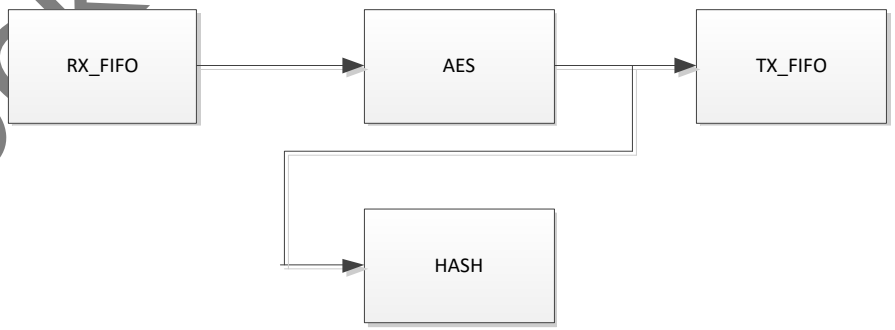

Fig.3-4 AES-HASH-TX mode

As shown in the figures above, we could facilitate operations in some cases. For example, secure boot, we need both AES and HASH operations for the same blocks of data. The data HASH gets from RX_FIFO or TX_FIFO is byteswaped if the byteswap function is configured.

3.4.6 Program Steps

The application could succeed various crypto operations if they program properly.

- Program the LLI address to DMA_LLI_ADDR;

- Program KEY, IV, or other parameter if needed;
- Program BC_CTL or HASH_CTL for control information;
- Prepare LLI Item;
- Enable interrupt, or do nothing;

All these operations could be in any order.

- Program DMA_CTL.start to start the operation;

This step should be the last configuration step. After this register is configured, other registers should not be changed.

- Wait interrupt asserted, or just poll the DMA_INT_ST bits;
- Program DMA_INT_ST to clear interrupt status, and get the result.

The application could also use LLI.pause when the next LLI item is not ready. After the new item is prepared, the application could program DMA_CTL.restart to continue previous operation.

Chapter 4 Digital Audio Codec

4.1 Overview

Digital Audio Codec is a 24-bit digital audio encoder/decoder which supports multiple sample rates. It is mainly composed of digital ADC and digital DAC. The function of digital ADC is to convert pulse density modulation format data into PCM format audio data through a series of filters and volume control. Decoding result of digital ADC is sent out through I2S/PCM interface. The aim of digital DAC is to process the data received from I2S/PCM interface through filters, volume control and modulation. Finally the modulated result is sent to Analog Codec.

The Digital Audio Codec supports the following features.

- Support 8-bit APB bus slave interface
- Support 3-channel digital ADC
- Support 1-channel digital DAC
- Support 2-channel I2S/PCM interface
- Support I2S/PCM master and slave mode
- Support 4-channel audio transmitting in I2S mode
- Support 2-channel audio receiving in I2S mode
- Support 2-channel audio transmitting or receiving in PCM mode
- Support I2S normal, left and right justified mode serial audio data transfer
- Support PCM early, late1, late2, late3 mode serial audio data transfer
- Support MSB or LSB first serial audio data transfer
- Support configurable SCLK and LRCK polarity
- Support 16 ~24 bit sample resolution for both digital ADC and digital DAC
- Support programmable left and right channel exchangeable in I2S mode and PCM mode for both digital ADC and digital DAC
- Support three modes of mixing for every digital DAC channel
- Both digital ADC and digital DAC support three groups of sample rates. Group 0 are 8khz/ 16khz/32kHz/64kHz/128khz, group 1 are 11.025khz/22.05khz/44.1khz/88.2khz/ 176.4khz and group 2 are 12khz/24khz/48khz/96khz/192khz
- Support digital ADC or digital DAC operating individually. The sample rate of digital ADC and digital DAC can be within any groups and any kind within one group
- Support digital ADC and digital DAC operating at the same time within same sample rate group. Sample rates of digital ADC and digital DAC are within the same group
- Support digital ADC and digital DAC operating at the same time within different sample rate group. Sample rate of digital ADC is within group 0 and digital DAC is within group 2 or digital ADC is within group 2 and digital DAC is within group 0, group 1 not supported
- The passband of digital ADC filters is 0.45625*fs
- Support digital ADC passband ripple within +/-0.1dB
- The stopband of digital ADC filters is 0.5*fs
- Support digital ADC stopband attenuation at least 60dB
- Support volume control for both digital ADC and digital DAC
- Support Automatic Level Control(ALC) and noise gate for digital ADC
- Support programmable negative and positive volume gain for both digital ADC and digital DAC
- Support communication with Analog Codec through I2C bus

4.2 Block Diagram

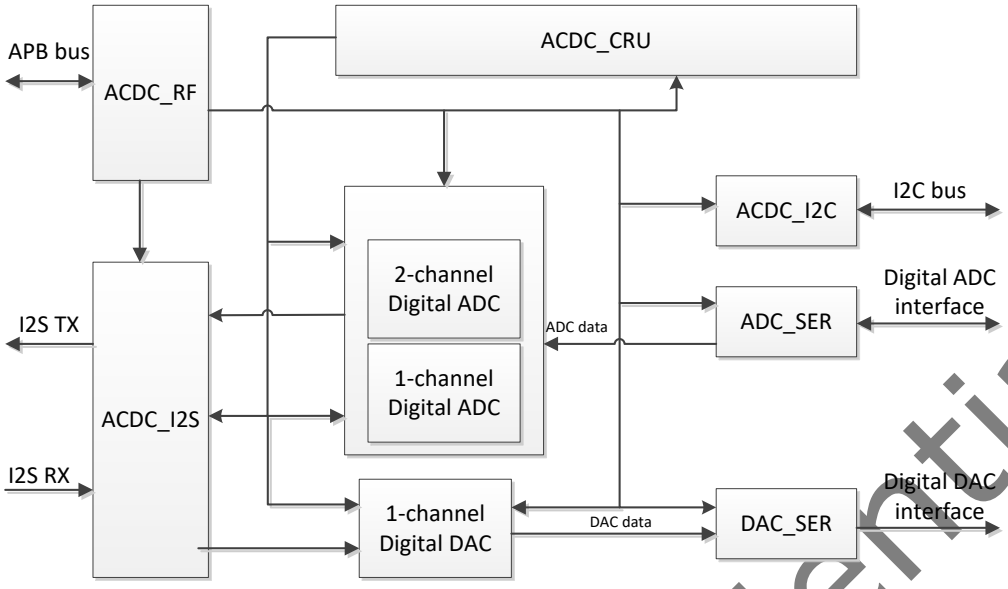

Fig. 4-1 Digital Audio Codec Block Diagram

ACDC_RF

The ACDC_RF implements the 8-bit APB slave operation through APB bus. It is responsible for configuring the operation registers of other modules.

ACDC_CRU

The ACDC_CRU implements clock and reset generation function. It is responsible for generating sample clock, digital ADC/DAC operation clock and I2S operation clock.

Digital ADC

There are 3 digital ADC channels in total inside Digital Audio Codec. The 2-channel digital ADC is composed of left channel and right channel while the 1-channel digital ADC contains only one channel. The digital ADC receives three channels data from ADC_SER module. It includes CIC filters, compensation filters, low-pass decimation filters, high-pass filters and other audio signal processing related modules. The output of digital ADC is sent to the I2S module.

ADC_SER

This module receives serial data from analog ADC. It extracts three channels from input serial data and distributes them to the corresponding digital ADC channel.

Digital DAC

There are 1 digital DAC channels inside the Audio Codec. The digital DAC receives audio data from I2S RX. It includes one CIC filter, one high-pass filter, several low-pass decimation filters, modulator and other audio signal processing related modules. The output of digital DAC is sent to DAC_SER module and serialized before transmitting to analog DAC.

DAC_SER

This module receives parallel data from digital DAC. It then serializes the parallel data and sends it to analog DAC.

ACDC_I2S

The I2S/PCM audio interface can be configured to master mode or slave mode. In Master Mode, SCLK and LRCK are configured as output. In Slave Mode, SCLK and LRCK are configured as input. The ACDC_I2S module can operate in TX or RX mode. When in TX mode, it receives audio data from digital ADC and sends it out through I2S TX interface. When in RX mode, it receives audio data from I2S RX interface and sends it to digital DAC.

ACDC_I2C

The ACDC_I2C module is used to communicate with analog ADC/DAC. It is responsible for sending 4-bit PGA gain codes for each channel to analog ADC/DAC every interval of time.

4.3 Function Description

The I2S/PCM interface of ACDCDIG is connected to the I2S1 controller. Please refers to the

I2S chapter for detailed information about I2S and PCM format that ACDCDIG supports.

4.3.1 Filters of Digital ADC

Digital Audio Codec receives 3 channels data such as DATA0, DATA1 and DATA2 from Analog Codec. DATA0 is connected to the left channel of 2-channel digital ADC, DATA1 is connected to the right channel of 2-channel digital ADC and DATA2 is connected to the 1-channel digital ADC.
In order to achieve PCM format audio data from DATA0, DATA1 and DATA2, a total of 8 filters for 2-channel digital ADC and 8 filters for 1-channel digital ADC are embedded. It includes a CIC decimation filter, a compensation filter, 4 half-band filters, a low-pass filter and a high-pass filter in 2-channel digital ADC or 1-channel digital ADC.
The CIC decimation filter achieves maximum 16-times decimation when in normal mode. It also can be programed to 8-times in low power mode 1 or 4-times decimation in low power mode 2. When operates in low power mode 1 or 2, the operating clock of digital ADC can be reduced to half or quarter of the normal mode. The problem is that signal indicators will become worse and some sample rates not support in low power mode 1 or 2. So make sure that the CIC decimation filter works in 16-times decimation unless you don't care about signal indicators.
Compensation filter is connected in series following CIC filter. Its function is to reduce the ripple of CIC filter output. It can be software enabled or disabled.
The 4 half-band filters and a low-pass filter each perform 2-times decimation. That means a maximum of 32-times decimation can be achieved. Not all of these 5 decimation filters are working all the time. How many of them are needed to work depends on the sample rate. For example, in order to output a 192K sample rate signal, only one filter works, the rest of filters are idle.
The high-pass filter is used to filter DC components in audio data stream. Result of high-pass filter is sent to the volume control.
The equivalent parameters of digital ADC filters are as follows.

Table 4-1 Equivalent parameters of digital ADC filters

Parameter	Test condition	Min	Typ	Max	Unit
passband	+/- 0.1dB	20	N/A	0.45625fs	Hz
passband ripple	N/A	N/A	N/A	+/- 0.1	dB
stopband	N/A	0.5fs	N/A	N/A	Hz
stopband attenuation	f>0.5fs	60	N/A	N/A	dB

4.3.2 Filters of Digital DAC

I2S module receives two-channel audio data from I2S RX interface and drives it to the digital DAC which supports mixing function. How to pour 2-channel audio data into 1-channel digital DAC can be achieved by programming mixing mode.
Audio processing of digital DAC is similar to that of digital ADC, which is almost the inverse process of digital ADC. The 1-channel digital DAC includes a high-pass filter and 5 half-band filters. The high-pass filter is used to filter DC components in audio data stream. Result of high-pass filter is sent to the digital DAC volume control module. The input of 5 half-band filters comes from output of volume control, each perform 2-times interpolation. But not all of them are working all the time. How many of them are needed to work depends on the sample rate of digital DAC. The result of half-band filters is sent to a third-order Sigma-Delta modulator.

4.3.3 Volume Control

PCM format data output from 3-channel digital ADC high-pass filters are fed into volume control module. The volume control module inside digital ADC contains several sub-modules such as noise gate, peak detect, frequency cross zero detect, ALC and digital gain control.

It can be digitally attenuated over a range of -96dB~0dB in 0.375dB/step for negative gain and amplified over a range of 0dB~96dB in 0.375dB/step for positive gain. Whether is attenuated or amplified can be software programed. The volume of each channel can be controlled separately. Each channel has an 8-bit register for volume control. For negative gain, 0xff corresponds to digital mute while 0x00 corresponds to 0dB. For positive gain, 0xff corresponds to maximum gain while 0x00 corresponds to 0dB.
As for digital DAC, output of high-pass filter is fed into volume control module. Similar to digital ADC, The volume control module inside digital DAC contains several sub-modules such as peak detect, frequency cross zero detect, LIMITER and digital gain control. It also can be digitally attenuated over a range of -96dB~0dB in 0.375dB/step for negative gain and amplified over a range of 0dB~96dB in 0.375dB/step for positive gain. Whether is attenuated or amplified can be software programed.

4.3.4 I2C Interface

The role of I2C inside the Digital Audio Codec is to send 4-bit PGA gain codes for each digital ADC channel to Analog Codec every interval of time. The interval of time is software programmable and the unit is the sample rate of digital ADC. There are 12 bits PGA gain codes in total for 3-channel digital ADC, requiring the I2C inside Digital Audio Codec to initiate a request to transmit two bytes when interval of time reaches. The I2C inside Digital Audio Codec can be enabled or disabled by software. Its output bus will multiplex with RKI2C2 controller.

4.4 Register Description

4.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
ACDCDIG_SYSCTRL0	0x0000	W	0x00000000	System Control Register
ACDCDIG_ADCVUCTL	0x0040	W	0x00000000	ADC Volume Control Register
ACDCDIG_ADCVUCTIME	0x0044	W	0x00000000	ADC Volume Control Time Limit Register
ACDCDIG_ADCDIGEN	0x0048	W	0x00000000	ADC Digital Enable Register
ACDCDIG_ADCCLKCTRL	0x004c	W	0x00000000	ADC Clock Control Register
ACDCDIG_ADCINT_DIV	0x0054	W	0x00000000	ADC Integer Clock Divider Register
ACDCDIG_ADCSCLKTXINT_DIV	0x006c	W	0x00000000	I2S SCLK TX Integer Divider Register
ACDCDIG_ADCCFG1	0x0084	W	0x00000000	ADC Configure Register 1
ACDCDIG_ADCVOLL0	0x0088	W	0x00000000	Volume of ADC Left Channel 0 Register
ACDCDIG_ADCVOLL1	0x008c	W	0x00000000	Volume of ADC Left Channel 2 Register
ACDCDIG_ADCVOLR0	0x0098	W	0x00000000	Volume of ADC right Channel 1 Register
ACDCDIG_ADCVOGP	0x00a8	W	0x00000000	ADC Volume Gain Polarity Register
ACDCDIG_ADCRVOLL0	0x00ac	W	0x000000ff	Internal Volume of ADC Left Channel 0 Register
ACDCDIG_ADCRVOLL1	0x00b0	W	0x000000ff	Internal Volume of ADC Left Channel 2 Register

Name	Offset	Size	Reset Value	Description
ACDCDIG_ADCRVOLR0	0x00bc	W	0x000000ff	Internal Volume of ADC right Channel 1 Register
ACDCDIG_ADCALC0	0x00cc	W	0x00000000	Automatic Level Control Register 0
ACDCDIG_ADCALC1	0x00d0	W	0x00000000	Automatic Level Control Register 1
ACDCDIG_ADCALC2	0x00d4	W	0x00000000	Automatic Level Control Register 2
ACDCDIG_ADCNG	0x00d8	W	0x00000000	ADC Noise Gate Control Register
ACDCDIG_ADCNGST	0x00dc	W	0x00000000	ADC Noise Gate Status Register
ACDCDIG_ADCHPFEN	0x00e0	W	0x00000000	ADC High-pass Filter Enable Register
ACDCDIG_ADCHPFCF	0x00e4	W	0x00000000	ADC High-pass Control Register
ACDCDIG_ADCPGL0	0x00ec	W	0x00000000	PGA Gain of Left Channel 0
ACDCDIG_ADCPGL1	0x00f0	W	0x00000000	PGA Gain of Left Channel 2
ACDCDIG_ADCPGR0	0x00fc	W	0x00000000	PGA Gain of right Channel 1
ACDCDIG_ADCLILMT1	0x010c	W	0x00000000	PGA Gain LIMITER Register 1
ACDCDIG_ADCLILMT2	0x0110	W	0x00000000	PGA Gain LIMITER Register 2
ACDCDIG_ADCDMICNG1	0x0114	W	0x00000000	Limiter Noise Gate Register 1
ACDCDIG_ADCDMICNG2	0x0118	W	0x00000000	LIMITER Noise Gate Register 2
ACDCDIG_DACVUCTL	0x0140	W	0x00000000	DAC Volume Control Register
ACDCDIG_DACVUCTIME	0x0144	W	0x00000000	DAC Volume Control Time Limit Register
ACDCDIG_DACDIGEN	0x0148	W	0x00000000	DAC Digital Enable Register
ACDCDIG_DACCLKCTRL	0x014c	W	0x00000000	DAC Clock Control Register
ACDCDIG_DACINT_DIV	0x0154	W	0x00000000	DAC Integer Clock Divider Register
ACDCDIG_DACSCLKRXINT_DIV	0x016c	W	0x0000001f	I2S SCLK RX Integer Divider Register
ACDCDIG_DACCFG1	0x0184	W	0x00000000	DAC Configure Register 1
ACDCDIG_DACMUTE	0x0188	W	0x00000000	DAC Mute Control Register
ACDCDIG_DACMUTEST	0x018c	W	0x00000000	DAC Mute Status Register
ACDCDIG_DACVOLL0	0x0190	W	0x00000000	Volume of DAC Left Channel 0 Register
ACDCDIG_DACVOGP	0x01b0	W	0x00000000	ADC Volume Gain Polarity Register
ACDCDIG_DACRVOLL0	0x01b4	W	0x000000ff	Internal Volume of DAC Left Channel 0 Register
ACDCDIG_DACLMT0	0x01d4	W	0x00000000	DAC LIMITER Register 0
ACDCDIG_DACLMT1	0x01d8	W	0x00000000	DAC LIMITER Register 1
ACDCDIG_DACLMT2	0x01dc	W	0x00000000	DAC LIMITER Register 2
ACDCDIG_DACMIXCTRLL	0x01e0	W	0x00000000	DAC Mixing Control Register Of Left Channels

Name	Offset	Size	Reset Value	Description
ACDCDIG_DACHPF	0x01e8	W	0x00000000	DAC High-pass Filter Control Register
ACDCDIG_I2C_FLT_CON0	0x0280	W	0x00000000	I2C Filter Control Register 0
ACDCDIG_I2C_FLT_CON1	0x0284	W	0x0000000f	I2C Filter Control Register 1
ACDCDIG_I2C_CON0	0x0288	W	0x00000000	I2C Control Register 0
ACDCDIG_I2C_CON1	0x028c	W	0x00000003	I2C Control Register 1
ACDCDIG_I2C_CLKDIVL0	0x0290	W	0x00000006	I2C Scl Low Level Divider Register 0
ACDCDIG_I2C_CLKDIVL1	0x0294	W	0x00000000	I2C Scl Low Level Divider Register 1
ACDCDIG_I2C_CLKDIVH0	0x0298	W	0x00000006	I2C Scl High Level Divider Register 0
ACDCDIG_I2C_CLKDIVH1	0x029c	W	0x00000000	I2C Scl High Level Divider Register 1
ACDCDIG_I2C_MAXCNT	0x02a0	W	0x00000080	I2C Master Transmit Count Register
ACDCDIG_I2C_SCLOE_DB0	0x02a4	W	0x00000020	Slave Hold Debounce Configure Register 0
ACDCDIG_I2C_SCLOE_DB1	0x02a8	W	0x00000000	Slave Hold Debounce Configure Register 1
ACDCDIG_I2C_SCLOE_DB2	0x02ac	W	0x00000020	Slave Hold Debounce Configure Register 2
ACDCDIG_I2C_SCLOE_DB3	0x02b0	W	0x00000020	Slave Hold Debounce Configure Register 3
ACDCDIG_I2C_TMOUTL	0x02b4	W	0x0000001f	I2C Transmit Request Time Out Register 0
ACDCDIG_I2C_TMOUTH	0x02b8	W	0x00000000	I2C Transmit Request Time Out Register 1
ACDCDIG_I2C_DEV_ADDR	0x02bc	W	0x00000048	I2C Slave Device Address Register
ACDCDIG_I2C_REG_ADDR	0x02c0	W	0x00000028	Register Address of I2C Slave
ACDCDIG_I2C_STATUS	0x02c4	W	0x00000000	I2C Status Register
ACDCDIG_I2S_TXCR0	0x0300	W	0x0000000f	Transmit Operation Control Register 0
ACDCDIG_I2S_TXCR1	0x0304	W	0x00000000	Transmit Operation Control Register 1
ACDCDIG_I2S_TXCR2	0x0308	W	0x00000000	Transmit Operation Control Register 2
ACDCDIG_I2S_RXCR0	0x030c	W	0x00000001	Receive Operation Control Register 0
ACDCDIG_I2S_RXCR1	0x0310	W	0x00000000	Receive Operation Control Register 1
ACDCDIG_I2S_CKR0	0x0314	W	0x00000000	Clock Generation Register 0

Name	Offset	Size	Reset Value	Description
ACDCDIG_I2S_CKR1	0x0318	W	0x00000000	Clock Generation Register 1
ACDCDIG_I2S_XFER	0x031c	W	0x00000000	Transfer Start Register
ACDCDIG_I2S_CLR	0x0320	W	0x00000000	SCLK Domain Logic Clear Register
ACDCDIG_VERSION	0x0380	W	0x00000002	Version Register

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

4.4.2 Detail Register Description

ACDCDIG_SYSCTRL0

Address: Operational Base + offset (0x0000)

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3	RW	0x0	glb_cke Global enable for synchronization signal of both Digital ADC and DAC. It works well when Digital ADC and DAC operate in synchronous mode where the Digital ADC and DAC share the same d2a_clk and d2a_sync to communicate with Analog ADC/DAC. If Digital ADC and DAC operate in asynchronous mode meaning that they are independent, glb_cke is ineffective. 1'b0: Disable 1'b1: Enable
2	RW	0x0	clk_sel Selects the operation clock(32.768/45.154/49.152MHz) generated by Digital ADC or by Digital DAC is connected to d2a_clk to communicate with Analogy ADC/DAC. 1'b0: Operation clock generated by Digital ADC is connected to d2a_clk. 1'b1: Operation clock generated by Digital DAC is connected to d2a_clk
1	RW	0x0	sync_sel Selects the synchronization signal generated by Digital ADC or by Digital DAC is connected to d2a_sync to communicate with Analogy ADC/DAC. 1'b0: Synchronization signal generated by Digital ADC is connected to d2a_sync. 1'b1: Synchronization signal generated by Digital DAC is connected to d2a_sync
0	RO	0x0	reserved

ACDCDIG_ADCVUCTL

Address: Operational Base + offset (0x0040)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	adc_byps ADC volume control bypass. 1'b0: ADC volume control enable 1'b1: ADC volume control bypass
1	RW	0x0	adc_fade ADC volume adjust mode. 1'b0: Update to new volume immediately 1'b1: Update volume as adc_zdt field describes
0	RW	0x0	adc_zdt ADC volume cross zero detect enable. It works when adc_byps is 1'b0 and adc_fade is 1'b1. 1'b0: Volume adjusts every sample 1'b1: Volume adjusts only when audio waveform crosses zero or volume-control time-limit condition meets.

ACDCDIG_ADCVUCTIME

Address: Operational Base + offset (0x0044)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	adc_vuct Volume control time limit, valid only in fade cross zero mode. Time limit = adc_vuct*(1/sample rate) Unit: Sample rate

ACDCDIG_ADCDIGEN

Address: Operational Base + offset (0x0048)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4	RW	0x0	adc_glben Global enable of all ADC channels. Only when adc_glben and the enable signal corresponding to each ADC channel is valid before starting work. 1'b0: Disable 1'b1: Enable
3:2	RO	0x0	reserved
1	RW	0x0	adcen_l2 ADC left channel 2 enable. 1'b0: Disable 1'b1: Enable
0	RW	0x0	adcen_l0r1 ADC left channel 0 and right channel 1 enable. 1'b0: Disable 1'b1: Enable

ACDCDIG_ADCCLKCTRL

Address: Operational Base + offset (0x004c)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:6	RW	0x0	cic_ds_ratio Cic filter decimation ratio 2'b00: 16 times decimation 2'b01: 8 times decimation other: 4 times decimation
5	RW	0x0	adc_cke ADC operation clock enable. 1'b0: Disable 1'b1: Enable
4	RW	0x0	i2stx_cke Clock enable of internal I2S TX channel. 1'b0: Disable 1'b1: Enable
3	RW	0x0	cke_bclktx Clock enable of sclk_out_tx. 1'b0: Disable 1'b1: Enable
2	RW	0x0	filter_gate_en Filter gate enable. There are some filters in ADC, they work depends on the sample rate. If any filters not work, the filter and its corresponding memory clock will be gated if filter_gate_en is 1'b1, otherwise the clock will be still active. 1'b0: Don't gate filters' clock. 1'b1: Gate filters' clock
1	RW	0x0	adc_sync_ena Enable of the synchronization signal generated by ADC. Note that only when both glb_cke and adc_sync_ena are equal to 1'b1, the synchronization signal of ADC can be generated. 1'b0: Disable 1'b1: Enable
0	RW	0x0	adc_sync_status There is a counter to generate synchronization signal of ADC. In order to ensure the integrity of synchronization signal, it is necessary to read back adc_sync_status to judge whether the counter stops working when adc_sync_ena is set from 1'b1 to 1'b0. If the signal is read back to 1'b0, it means that the counter stops working and the synchronization signal of ADC is complete.

ACDCDIG_ADCINT_DIV

Address: Operational Base + offset (0x0054)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	int_div_con Interger clock divider to provide 6.144/5.644/4.096MHz sample clock for internal filters. Make sure that int_div_con is an odd number between 7(8 times division) and 15(16 times division).

ACDCDIG_ADCSCLKTXINT_DIV

Address: Operational Base + offset (0x006c)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	scktxdiv Integer clock divider to generate sclk_out_tx when I2S TX works in master mode. It is ignored when in slave mode.

ACDCDIG_ADCCFG1

Address: Operational Base + offset (0x0084)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:2	RW	0x0	adcsrt Sample rates of ADC 3'b000: 12kHz/11.024kHz/8kHz 3'b001: 24kHz/22.05kHz/16kHz 3'b010: 32kHz 3'b011: 48kHz/44.1kHz 3'b100: 96kHz/88.2kHz/64kHz 3'b101~3'b111: 192kHz/176.4kHz/128kHz
1	RW	0x0	sig_scale_mode Signal scale mode select 1'b0: Cic output the normal latitude 1'b1: Scale the cic output to half of the normal latitude and scale 2 times after high-pass filter.
0	RW	0x0	fir_com_bps FIR compensate filter bypass control 1'b0: Not bypass 1'b1: Bypass

ACDCDIG_ADCVOLL0

Address: Operational Base + offset (0x0088)

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

Bit	Attr	Reset Value	Description
7:0	RW	0x00	adclv0 Volume of ADC left channel 0 0db~-96db, 0.375db/step. 8'h0: 0db 8'h1: -0.375db 8'h2: -0.75db 8'h3: -1.125db … 8'hff: -96db

ACDCDIG_ADCVOLL1

Address: Operational Base + offset (0x008c)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	adclv1 Volume of ADC left channel 2 0db~-96db, 0.375db/step. 8'h0: 0db 8'h1: -0.375db 8'h2: -0.75db 8'h3: -1.125db … 8'hff: -96db

ACDCDIG_ADCVOLR0

Address: Operational Base + offset (0x0098)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	adcrv0 Volume of ADC right channel 1 0db~-96db, 0.375db/step 8'h0: 0db 8'h1: -0.375db 8'h2: -0.75db 8'h3: -1.125db … 8'hff: -96db

ACDCDIG_ADCVOGP

Address: Operational Base + offset (0x00a8)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved

Bit	Attr	Reset Value	Description
2	RW	0x0	volgpl2 Gain polarity for the volume of ADC left channel 2 1'b0: Negative gain 1'b1: Positive gain
1	RW	0x0	volgpr1 Gain polarity for the volume of ADC right channel 1 1'b0: Negative gain 1'b1: Positive gain
0	RW	0x0	volgpl0 Gain polarity for the volume of ADC left channel 0 1'b0: Negative gain 1'b1: Positive gain

ACDCDIG_ADCRVOLL0

Address: Operational Base + offset (0x00ac)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0xff	rvoll0 Internal real-time volume of ADC left channel 0

ACDCDIG_ADCRVOLL1

Address: Operational Base + offset (0x00b0)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0xff	rvoll1 Internal real-time volume of ADC left channel 2

ACDCDIG_ADCRVOLR0

Address: Operational Base + offset (0x00bc)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0xff	rvolr0 Internal real-time volume of ADC right channel 1

ACDCDIG_ADCALC0

Address: Operational Base + offset (0x00cc)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	alcl2 Automatic level control enable for ADC left channel 2. 1'b0: Disable 1'b1: Enable

Bit	Attr	Reset Value	Description
1	RW	0x0	alcr1 Automatic level control enable for ADC right channel 1. 1'b0: Disable 1'b1: Enable
0	RW	0x0	alcl0 Automatic level control enable for ADC left channel 0. 1'b0: Disable 1'b1: Enable

ACDCDIG_ADCALC1

Address: Operational Base + offset (0x00d0)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:4	RW	0x0	alcarate ALC attack rate=sample rate/(8*power(2,alcarate))
3:0	RW	0x0	alcrrate ALC release rate=sample rate/(8*power(2,alcrrate))

ACDCDIG_ADCALC2

Address: Operational Base + offset (0x00d4)

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6:4	RW	0x0	alcmax The highest threshold of ALC 3'b000~3'b100: 0db~-12db, 3db/step 3'b101~3'b111: -18db~-30db, 6db/step
3	RO	0x0	reserved
2:0	RW	0x0	alcmin The lowest threshold of ALC 3'b000~3'b100: 0db~-12db, 3db/step 3'b101~3'b111: -18db~-30db, 6db/step

ACDCDIG_ADCNG

Address: Operational Base + offset (0x00d8)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	ngchl Noise gate channel 1'b0: Individual channel(or) 1'b1: Both channel(and)
6	RW	0x0	ngen Noise gate enable 1'b0: Noise gate disable 1'b1: Noise gate enable

Bit	Attr	Reset Value	Description
5	RW	0x0	ngboost Noise gate boost 1'b0: Normal noise gate 1'b1: Boost noise gate
4:2	RW	0x0	nggate Noise gate threshold If ngboost is 1'b0: 3'b000~3'b111: -63db~-84db, 3db/step If ngboost is 1'b1: 3'b000~3'b111: -33db~-54db, 3db/step
1:0	RW	0x0	ngdly Noise gate delay. The delay time before the noise gate attacks. 2'b00: 2048 samples 2'b01: 4096 samples 2'b10: 8192 samples 2'b11: 16384 samples

ACDCDIG_ADCNGST

Address: Operational Base + offset (0x00dc)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RO	0x0	ngstl2 Noise gates valid status of left channel 2. 1'b0: Not in NG status 1'b1: Now in NG status
0	RO	0x0	ngstl0r1 Noise gates valid status of left channel 0 and right channel 1. 1'b0: Not in NG status 1'b1: Now in NG status

ACDCDIG_ADCHPFEN

Address: Operational Base + offset (0x00e0)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	hpfen_l2 High-pass filter enable for left channel 2 1'b0: High-pass filter is disabled 1'b1: High-pass filter is enabled
1	RW	0x0	hpfen_r1 High-pass filter enable for right channel 1 1'b0: High-pass filter is disabled 1'b1: High-pass filter is enabled

Bit	Attr	Reset Value	Description
0	RW	0x0	hpfen_l0 High-pass filter enable for left channel 0 1'b0: High-pass filter is disabled 1'b1: High-pass filter is enabled

ACDCDIG_ADCHPFCF

Address: Operational Base + offset (0x00e4)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1:0	RW	0x0	hpfcf High-pass filter control 2'b00: 3.79Hz 2'b01: 60Hz 2'b10: 243Hz 2'b11: 493Hz

ACDCDIG_ADCPGL0

Address: Operational Base + offset (0x00ec)

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3:0	RW	0x0	pga_l0 PGA gain of left channel 0 for analogy ADC Minimal gain: -18dB, maximum gain: 27dB, step: 3dB. The minimal gain corresponds to 4'b0000 and the maximum gain corresponds to 4'b1111

ACDCDIG_ADCPGL1

Address: Operational Base + offset (0x00f0)

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3:0	RW	0x0	pga_l1 PGA gain of left channel 2 for analogy ADC Minimal gain: -18dB, maximum gain: 27dB, step: 3dB. The minimal gain corresponds to 4'b0000 and the maximum gain corresponds to 4'b1111

ACDCDIG_ADCPGR0

Address: Operational Base + offset (0x00fc)

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved

Bit	Attr	Reset Value	Description
3:0	RW	0x0	pga_r0 PGA gain of right channel 1 for analogy ADC Minimal gain: -18dB, maximum gain: 27dB, step: 3dB. The minimal gain corresponds to 4'b0000 and the maximum gain corresponds to 4'b1111

ACDCDIG_ADCLILMT1

Address: Operational Base + offset (0x010c)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	lmt_en PGA gain LIMITER enable 1'b0: Disable 1'b1: Enable
6:4	RW	0x0	max_lilmt The highest threshold of LIMITER 3'b000~3'b100: 0db~-12db, 3db/step 3'b101~3'b111: -18db~-30db, 6db/step
3	RO	0x0	reserved
2:0	RW	0x0	min_lilmt The lowest threshold of LIMITER 3'b000~3'b100: 0db~-12db, 3db/step 3'b101~3'b111: -18db~-30db, 6db/step

ACDCDIG_ADCLILMT2

Address: Operational Base + offset (0x0110)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:4	RW	0x0	atk_rate LIMITER attack rate=(power(2, atk_rate)*(8*clk)), clk is such as 4.096Mhz, 5.6448Mhz, 6.144Mhz
3:0	RW	0x0	rls_rate LIMITER release rate=(power(2, rls_rate)*(8*clk)), clk is such as 4.096Mhz, 5.6448Mhz, 6.144Mhz

ACDCDIG_ADCDMICNG1

Address: Operational Base + offset (0x0114)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	ngchl_li Noise gate channel 1'b0: Individual channel(or) 1'b1: Both channel(and)

Bit	Attr	Reset Value	Description
6	RW	0x0	ngen_li Noise gate enable 1'b0: Noise gate disable 1'b1: Noise gate enable
5	RW	0x0	ngboost_li Noise gate boost 1'b0: Normal noise gate 1'b1: Boost noise gate
4:2	RW	0x0	nggate_li Noise gate threshold If ngboost is 1'b0: 3'b000~3'b111: -63db~-84db, 3db/step If ngboost is 1'b1: 3'b000~3'b111: -33db~-54db, 3db/step
1:0	RW	0x0	ngdly_li Noise gate delay. The delay time before the noise gate attacks. 2'b00: 2048 samples 2'b01: 4096 samples 2'b10: 8192 samples 2'b11: 16384 samples

ACDCDIG_ADCDMICNG2

Address: Operational Base + offset (0x0118)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RO	0x0	ngvalid_li_l2 Noise gates valid status of Limiter for left channel 2. 1'b0: Not in NG status 1'b1: Now in NG status
0	RO	0x0	ngvalid_li_l0r1 Noise gates valid status of Limiter for left channel 0 and right channel 1. 1'b0: Not in NG status 1'b1: Now in NG status

ACDCDIG_DACVUCTL

Address: Operational Base + offset (0x0140)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	dac_byps DAC volume control bypass 1'b0: DAC volume control enable 1'b1: DAC volume control bypass
1	RW	0x0	dac_fade DAC volume adjust mode 1'b0: Update to new volume immediately. 1'b1: Update volume as dac_zdt field describes.
0	RW	0x0	dac_zdt DAC volume cross zero detect enable. It works when dac_byps is 1'b0 and dac_fade is 1'b1. 1'b0: Volume adjusts every sample. 1'b1: Volume adjusts only when audio waveform crosses zero or volume-control time-limit condition meets.

ACDCDIG_DACVUCTIME

Address: Operational Base + offset (0x0144)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	dac_vuct Volume control time limit, valid only in fade cross zero mode. Time limit = dac_vuct*(1/sample rate) Unit: Sample rate

ACDCDIG_DACDIGEN

Address: Operational Base + offset (0x0148)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4	RW	0x0	adc_glben Global enable of all DAC channels. Only when adc_glben and the enable signal corresponding to each DAC channel is valid before starting work. 1'b0: Disable 1'b1: Enable
3:1	RO	0x0	reserved
0	RW	0x0	dacen_l0 DAC left channel 0 enable 1'b0: Disable 1'b1: Enable

ACDCDIG_DACCLKCTRL

Address: Operational Base + offset (0x014c)

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5	RW	0x0	dac_cke DAC operation clock enable 1'b0: Disable 1'b1: Enable
4	RW	0x0	i2srx_cke Clock enable of internal I2S RX channel 1'b0: Disable 1'b1: Enable
3	RW	0x0	cke_bclkrx Clock enable of sclk_out_rx 1'b0: Disable 1'b1: Enable
2	RW	0x0	dac_sync_ena Enable of the synchronization signal generated by DAC. Note that only when both glb_cke and dac_sync_ena are equal to 1'b1, the synchronization signal of DAC can be generated. 1'b0: Disable 1'b1: Enable
1	RW	0x0	dac_sync_status There is a counter to generate synchronization signal of DAC. In order to ensure the integrity of synchronization signal, it is necessary to read back dac_sync_status to judge whether the counter stops working when dac_sync_ena is set from 1'b1 to 1'b0. If the signal is read back to 1'b0, it means that the counter stops working and the synchronization signal of DAC is complete.
0	RO	0x0	reserved

ACDCDIG_DACINT_DIV

Address: Operational Base + offset (0x0154)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	int_div_con Interger clock divider to provide 6.144/5.644/4.096MHz sample clock for internal filters. Make sure that int_div_con is an odd number between 7(8 times division) and 15(16 times division).

ACDCDIG_DACSCLKRXINT_DIV

Address: Operational Base + offset (0x016c)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x1f	sckrxdiv Interger clock divider to generate sclk_out_rx when I2S RX works in master mode. It is ignored when in slave mode.

ACDCDIG_DACCFG1

Address: Operational Base + offset (0x0184)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:2	RW	0x0	dacsrt Sample rates of DAC 3'b000: 12kHz/11.024kHz/8kHz 3'b001: 24kHz/22.05kHz/16kHz 3'b010: 32kHz/48kHz/44.1kHz 3'b011: 96kHz/88.2kHz/64kHz 3'b100: 192kHz/176.4kHz/128kHz 3'b101~3'b111: Reserved
1:0	RO	0x0	reserved

ACDCDIG_DACMUTE

Address: Operational Base + offset (0x0188)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	dacunmt 1'b0: DAC normal mode 1'b1: DAC unmute mode. In this mode, DAC volume control block will adjust volume to match the value in DACVOLL* and DACVOLR*.
0	RW	0x0	dacmt 1'b0: DAC normal mode 1'b1: DAC mute mode

ACDCDIG_DACMUTEST

Address: Operational Base + offset (0x018c)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4	RO	0x0	unmutest_l0 Unmute status for DAC left channel 0. When unmute is done, it indicates that internal volume is equal to the value programmed in DACVOLL* and DACVOLR*. 1'b0: Unmute not done 1'b1: Unmute done
3:1	RO	0x0	reserved
0	RO	0x0	mutest_l0 Mute status for DAC left channel 0 1'b0: Not mute 1'b1: Mute

ACDCDIG_DACVOLL0

Address: Operational Base + offset (0x0190)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	daclv0 Volume of DAC left channel 0 0db~-96db, 0.375db/step. 8'h0: 0db 8'h1: -0.375db 8'h2: -0.75db 8'h3: -1.125db … 8'hff: -96db

ACDCDIG_DACVOGP

Address: Operational Base + offset (0x01b0)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	volgpl0 Gain polarity for the volume of DAC left channel 0 1'b0: negative gain 1'b1: positive gain

ACDCDIG_DACRVOLL0

Address: Operational Base + offset (0x01b4)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0xff	rvoll0 Internal real-time volume of DAC left channel 0

ACDCDIG_DACLMT0

Address: Operational Base + offset (0x01d4)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	limen Limiter enable 1'b0: Disable 1'b1: Enable
0	RW	0x0	limdct LIMITER detect mode 1'b0: (Left channel + right channel)/2 1'b0: Left channel or right channel independently

ACDCDIG_DACLMT1

Address: Operational Base + offset (0x01d8)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:4	RW	0x0	atk_rate LIMITER attack rate=(power(2, atk_rate)*(8*clk)), clk is such as 4.096Mhz, 5.6448Mhz, 6.144Mhz.
3:0	RW	0x0	rls_rate LIMITER release rate=(power(2, rls_rate)*(8*clk)), clk is such as 4.096Mhz, 5.6448Mhz, 6.144Mhz.

ACDCDIG_DACLMT2

Address: Operational Base + offset (0x01dc)

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6:4	RW	0x0	max_lilmt The highest threshold of LIMITER 3'b000~3'b100: 0db~-12db, 3db/step 3'b101~3'b111: -18db~-30db, 6db/step
3	RO	0x0	reserved
2:0	RW	0x0	min_lilmt The lowest threshold of LIMITER 3'b000~3'b100: 0db~-12db, 3db/step 3'b101~3'b111: -18db~-30db, 6db/step

ACDCDIG_DACMIXCTRLL

Address: Operational Base + offset (0x01e0)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1:0	RW	0x0	mixmode_l0 DAC left channel 0 mixing mode 2'b00: Left channel 2'b01: Right channel 2'b10~2'b11: (Left channel + right channel)/2

ACDCDIG_DACHPF

Address: Operational Base + offset (0x01e8)

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5:4	RW	0x0	hpfcf High-pass filter control 2'b00: 80Hz 2'b01: 100Hz 2'b10: 120Hz 2'b11: 140Hz
3:1	RO	0x0	reserved

Bit	Attr	Reset Value	Description
0	RW	0x0	hpfen_l0 High-pass filter enable for left channel 0 1'b0: High-pass filter is disabled. 1'b1: High-pass filter is enabled

ACDCDIG_I2C_FLT_CON0

Address: Operational Base + offset (0x0280)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:4	RW	0x0	flt_r Filter scl rising edge glitches of width less than flt_r * Tclk_i2c
3:0	RW	0x0	flt_f Filter scl falling edge glitches of width less than flt_f * Tclk_i2c

ACDCDIG_I2C_FLT_CON1

Address: Operational Base + offset (0x0284)

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	h0_check_scl 1'b0: Check if scl been pull down by slave at the whole SCL_HIGH. 1'b1: Check if scl been pull down by slave only at the h0 of SCL_HIGH(SCL_HIGH including h0~h7).
5	RW	0x0	nak_release_scl 1'b0: Hold scl as low when received nack. 1'b1: Release scl as high when received nack
4	RW	0x0	flt_en SCL edge glitch filter enable 1'b0: Disable 1'b1: Enable
3:0	RW	0xf	slv_hold_scl_th Slave hold scl threshold = slv_hold_scl_db * Tclk_i2c

ACDCDIG_I2C_CON0

Address: Operational Base + offset (0x0288)

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	act2nak Operation when NAK handshake is received. 1'b0: Ignored 1'b1: Stop transaction
5:3	RO	0x0	reserved

Bit	Attr	Reset Value	Description
2:1	RW	0x0	i2c_mode I2C mode select 2'b00: Transmit only 2'b01~2'b11: Reserved
0	RW	0x0	i2c_en I2C enable 1'b0: Not enable 1'b1: Enable

ACDCDIG_I2C_CON1

Address: Operational Base + offset (0x028c)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:6	RW	0x0	stop_setup Stop setup configure TSU: Sto = (stop_setup + 1) * T(SCL_HIGH) + Tclk_i2c
5:4	RW	0x0	start_setup Start setup configure TSU: Sta = (start_setup + 1) * T(SCL_HIGH) + Tclk_i2c THD: Sta = (start_setup + 2) * T(SCL_HIGH) - Tclk_i2c
3:0	RW	0x3	data_upd_st SDA update point configure. Used to configure sda change state when scl is low, used to adjust setup/hold time. 4'bn: Thold = (n + 1) * Tclk_i2c Note: 0 <= n <= 5

ACDCDIG_I2C_CLKDIVL0

Address: Operational Base + offset (0x0290)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x06	clkdivl0 Low 8 bits of scl low level clock divider. The value of 16 bits scl low level clock divider updates only when clkdivl1 is programmed. So ensure to program clkdivl0 firstly, followed by clkdivl1.

ACDCDIG_I2C_CLKDIVL1

Address: Operational Base + offset (0x0294)

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

Bit	Attr	Reset Value	Description
7:0	RW	0x00	clkdivl1 High 8 bits of scl low level clock divider. The value of 16 bits scl low level clock divider updates only when clkdivl1 is programmed. So ensure to program clkdivl0 firstly, followed by clkdivl1. T(SCL_LOW) = Tclk_i2c * (CLKDIVL + 1) * 8, where the CLKDIVL is equal to clkdivl0+clkdivl1*256.

ACDCDIG_I2C_CLKDIVH0

Address: Operational Base + offset (0x0298)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x06	clkdivh0 Low 8 bits of scl high level clock divider. The value of 16 bits scl high level clock divider updates only when clkdivh1 is programmed. So ensure to program clkdivh0 firstly, followed by clkdivh1.

ACDCDIG_I2C_CLKDIVH1

Address: Operational Base + offset (0x029c)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	clkdivh1 High 8 bits of scl high level clock divider. The value of 16 bits scl high level clock divider updates only when clkdivh1 is programmed. So ensure to program clkdivh0 firstly, followed by clkdivh1. T(SCL_LOW) = Tclk_i2c * (CLKDIVH + 1) * 8, where the CLKDIVH is equal to clkdivh0+clkdivh1*256.

ACDCDIG_I2C_MAXCNT

Address: Operational Base + offset (0x02a0)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RO	0x1	idle 1'b0: I2C module is busy. 1'b1: I2C module is idle
6	RO	0x0	reserved
5:0	RW	0x00	mtxcnt Master transmit number Specify the total bytes to be transmitted (0~32).

ACDCDIG_I2C_SCLOE_DB0

Address: Operational Base + offset (0x02a4)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x20	scloedb0 Bit7~bit0 of slave hold scl debounce register Cycles for debounce (unit: Tclk_i2c)

ACDCDIG_I2C_SCLOE_DB1

Address: Operational Base + offset (0x02a8)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	scloedb1 Bit15~bit8 of slave hold scl debounce register Cycles for debounce (unit: Tclk_i2c)

ACDCDIG_I2C_SCLOE_DB2

Address: Operational Base + offset (0x02ac)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x20	scloedb2 Bit23~bit16 of slave hold scl debounce register Cycles for debounce (unit: Tclk_i2c)

ACDCDIG_I2C_SCLOE_DB3

Address: Operational Base + offset (0x02b0)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x20	scloedb3 Bit31~bit24 of slave hold scl debounce register Cycles for debounce (unit: Tclk_i2c)

ACDCDIG_I2C_TMOUTL

Address: Operational Base + offset (0x02b4)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x1f	tmoutl Low 8 bits of 16 bits I2C transmit request time out register. The value of 16 bits I2C transmit request time out register updates only when tmouth is programmed. So ensure to program tmoutl firstly, followed by tmouth.

ACDCDIG_I2C_TMOUTH

Address: Operational Base + offset (0x02b8)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	tmouth High 8 bits of 16 bits I2C transmit request time out register. The value of 16 bits I2C transmit request time out register updates only when tmouth is programmed. So ensure to program tmoutl firstly, followed by tmouth.

ACDCDIG_I2C_DEV_ADDR

Address: Operational Base + offset (0x02bc)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x48	dev_addr I2C slave device address

ACDCDIG_I2C_REG_ADDR

Address: Operational Base + offset (0x02c0)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x28	regaddr I2C slave register address to be accessed

ACDCDIG_I2C_STATUS

Address: Operational Base + offset (0x02c4)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	slavehdsclst Slave hold scl status bit 1'b0: Slave not hold scl 1'b1: Slave hold scl, write 1'b1 to clear.
6	RW	0x0	nakrcvst NAK handshake received status bit 1'b0: No NAK handshake received. 1'b1: NAK handshake received, write 1'b1 to clear.
5	RW	0x0	stopst Stop operation finished status bit 1'b0: No finished 1'b1: Stop operation finished, write 1'b1 to clear.
4	RW	0x0	startst Start operation finished status bit 1'b0: Not finished 1'b1: Start operation finished, write 1'b1 to clear.
3	RO	0x0	reserved

Bit	Attr	Reset Value	Description
2	RW	0x0	mbtfst I2C_MTXCNT data transfer finished status bit 1'b0: Not finished 1'b1: I2C_MTXCNT data transfer finished, write 1'b1 to clear.
1	RO	0x0	reserved
0	RW	0x0	btfst Byte tx finished status bit 1'b0: Byte tx not finish 1'b1: Byte tx finished, write 1'b1 to clear.

ACDCDIG_I2S_TXCR0

Address: Operational Base + offset (0x0300)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:6	RW	0x0	pbm 2'b00: PCM no delay mode 2'b01: PCM delay 1 mode 2'b10: PCM delay 2 mode 2'b11: PCM delay 3 mode
5	RW	0x0	tfs 1'b0: I2S format 1'b1: PCM format
4:0	RW	0x0f	vdw 5'b00000~5'b01110: Reserved 5'b01111: 16bit 5'b10000: 17bit 5'b10001: 18bit 5'b10010: 19bit …… 5'b11100: 29bit 5'b11101: 30bit 5'b11110: 31bit 5'b11111: 32bit

ACDCDIG_I2S_TXCR1

Address: Operational Base + offset (0x0304)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:6	RW	0x0	tcsr 2'b00: Two channel 2'b01: Four channel 2'b10~2'b11: Reserved
5	RO	0x0	reserved

Bit	Attr	Reset Value	Description
4	RW	0x0	cex Exchange left channel and right channel in the every transmit line. 1'b0: Not exchange 1'b1: Exchange
3	RO	0x0	reserved
2	RW	0x0	fbm 1'b0: MSB 1'b1: LSB
1:0	RW	0x0	ibm 2'b00: I2S normal 2'b01: I2S left justified 2'b10: I2S right justified 2'b11: Reserved

ACDCDIG_I2S_TXCR2

Address: Operational Base + offset (0x0308)

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5:0	RW	0x00	rcnt Only valid in I2S Right justified format and slave tx mode is selected. Start to transmit data rcnt sclk cycles after left channel valid.

ACDCDIG_I2S_RXCR0

Address: Operational Base + offset (0x030c)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:6	RW	0x0	pbm 2'b00: PCM no delay mode 2'b01: PCM delay 1 mode 2'b10: PCM delay 2 mode 2'b11: PCM delay 3 mode
5	RW	0x0	tfs 1'b0: I2S format 1'b1: PCM format

Bit	Attr	Reset Value	Description
4:0	RW	0x01	vdw 5'b00000~5'b01110: Reserved 5'b01111: 16bit 5'b10000: 17bit 5'b10001: 18bit 5'b10010: 19bit …… 5'b11100: 29bit 5'b11101: 30bit 5'b11110: 31bit 5'b11111: 32bit

ACDCDIG_I2S_RXCR1

Address: Operational Base + offset (0x0310)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:6	RW	0x0	rcsr 2'b00: Two channel 2'b01~2'b11: Reserved
5	RO	0x0	reserved
4	RW	0x0	cex Exchange left channel and right channel in the every receive line. 1'b0: Not exchange 1'b1: Exchange
3	RO	0x0	reserved
2	RW	0x0	fbm 1'b0: MSB 1'b1: LSB
1:0	RW	0x0	ibm 2'b00: I2S normal 2'b01: I2S left justified 2'b10: I2S right justified 2'b11: Reserved

ACDCDIG_I2S_CKR0

Address: Operational Base + offset (0x0314)

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3:2	RW	0x0	rsd I2S rx sclk divider for rx_lrck generator 2'b00: 64 2'b01: 128 2'b10~2'b11: 256

Bit	Attr	Reset Value	Description
1:0	RW	0x0	tsd I2S tx sclk divider for tx_lrck generator 2'b00: 64 2'b01: 128 2'b10~2'b11: 256

ACDCDIG_I2S_CKR1

Address: Operational Base + offset (0x0318)

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3	RW	0x0	mss 1'b0: Master mode(sclk output) 1'b1: Slave mode(sclk input)
2	RW	0x0	ckp 1'b0: Sample data at posedge sclk and drive data at negedge sclk 1'b1: Sample data at negedge sclk and drive data at posedge sclk
1	RW	0x0	rlp 1'b0: Normal polarity (I2S normal: Low for left channel, high for right channel I2S left/right just: High for left channel, low for right channel PCM start signal: High valid) 1'b1: Opposite polarity (I2S normal: High for left channel, low for right channel I2S left/right just: Low for left channel, high for right channel PCM start signal: Low valid)
0	RW	0x0	tlp 1'b0: Normal polarity (I2S normal: Low for left channel, high for right channel I2S left/right just: High for left channel, low for right channel PCM start signal: High valid) 1'b1: Opposite polarity (I2S normal: High for left channel, low for right channel I2S left/right just: Low for left channel, high for right channel PCM start signal: Low valid)

ACDCDIG_I2S_XFER

Address: Operational Base + offset (0x031c)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	rxs 1'b0: Stop RX transfer. 1'b1: Start RX transfer
0	RW	0x0	txs 1'b0: Stop TX transfer. 1'b1: Start TX transfer

ACDCDIG_I2S_CLR

Address: Operational Base + offset (0x0320)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	W1C	0x0	rxc This is a self-cleared bit. Write 1 to clear all receive logic.
0	W1C	0x0	txc This is a self-cleared bit. Write 1 to clear all transmit logic.

ACDCDIG_VERSION

Address: Operational Base + offset (0x0380)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0x02	ver Version of ACDCDIG

4.5 Interface Description

Digital Audio Codec is connected to two groups of IO interfaces to communicate with Analog Codec. The following table shows the Digital Audio Codec group 0 interface description.

Table 4-2 Digital Audio Codec Group 0 Interface Description

Module Pin	Direction	Pad Name	IOMUX Setting
D2A_CLK	O	CODEC_CLK_M0/PWM4_M1/TKEY0_M1/UART1_RX_M2/TKEY_DRIVE_M4/I2C0_SDA_M3/PWM_AUDIO_L_M2/GPIO0_D0_u	GRF_GPIO0D_IOMUX_L[3:0]=4'b0001
D2A_SYNC	O	CODEC_SYNC_M0/PWM5_M1/TKEY1_M1/UART1_TX_M2/SPI1_CS0n_M2/I2C0_SCL_M3/PWM_AUDIO_R_M2/GPIO0_D1_u	GRF_GPIO0D_IOMUX_L[7:4]=4'b0001
A2D_ADC_DATA	I	CODEC_ADC_D_M0/PWM6_M1/TKEY2_M1/UART2_CTSN_M0/SPI1_CLK_M2/I2S_MCLK_M1/GPIO0_D2_u	GRF_GPIO0D_IOMUX_L[11:8]=4'b0001
D2A_DAC_DATA	O	CODEC_DAC_DL_M0/PWM8/TKEY3_M1/UART2_RTSN_M0/SPI1_MOSI_M2/I2S_SCLK_TX_M1/GPIO0_D3_u	GRF_GPIO0D_IOMUX_L[15:12]=4'b0001

The following table shows the Digital Audio Codec group 1 interface description.

Table 4-3 Digital Audio Codec Group 1 Interface Description

Module Pin	Direction	Pad Name	IOMUX Setting
D2A_CLK	O	LCD_RDN/CIF_CLKOUT/UART1_CTSN_M1/TKEY16/PMU_SLEEP/PMU_STATE2/CODEC_CLK_M1/AONJTAG_TRSTn/DSPJTAG_TRSTn/GPIO0_A4_u	GRF_GPIO0A_IOMUX_H[3:0]=4'b0111
D2A_SYNC	O	LCD_WRN/CIF_CLKIN/UART1_RTSN_M1/PDM_CLK_M0/TKEY17/PMU_STATE3/CODEC_SYNC_M1/GPIO0_A5_d	GRF_GPIO0A_IOMUX_H[7:4]=4'b0001

Module Pin	Direction	Pad Name	IOMUX Setting
A2D_ADC_DATA	I	LCD_D2/CIF_D2/UART1_RX_M1/PDM_SDI_M0/TKEY18/PMU_STATE4/CODEC_ADC_D_M1/GPIO0_A6_d	GRF_GPIO0A_IOMUX_H[11:8]=4'b0111
D2A_DAC_DATA	O	LCD_D3/CIF_D3/UART1_TX_M1/PDM_CLK_S_M0/TKEY19/TEST_CLKOUT/CODEC_DAC_DL_M1/GPIO0_A7_d	GRF_GPIO0A_IOMUX_H[15:12]=4'b0111

4.6 Application Notes

4.6.1 Frequency Configuration

ACDC_CLK is input clock of Digital Audio Codec. Operation clock of digital ADC, DAC and I2S master mode are generated from ACDC_CLK. ACDC_CLK must be homologous to the MCLK of I2S master when the I2S module inside Digital Audio Codec acts as a slave. There is another clock named D2A_CLK acting as interface clock generated from ACDC_CLK and sent out to Analog Codec. In order to transfer audio data between digital Codec and Analog Codec, in addition to the audio data lines of digital ADC and DAC, a periodic synchronous signal named D2A_SYNC is needed to indicate the start of every sample data.
The relationship of ACDC_CLK, D2A_CLK, D2A_SYNC and sample rates is as follows where the ACDC_CLK is 8 times of D2A_SYNC.

Table 4-4 Relationship of ACDC_CLK, D2A_CLK, D2A_SYNC and Sample Rates in Normal Mode

ACDC_CLK	D2A_CLK	D2A_SYNC	Sample rates supported
49.152MHz	49.152MHz	6.144MHz	12/24/48/96/192kHz
45.154MHz	45.154MHz	5.644MHz	11.024/22.05/44.1/88.2/176.4 kHz
32.768MHz	32.768MHz	4.096MHz	8/16/32/64/128kHz

Table 4-5 Relationship of ACDC_CLK, D2A_CLK, D2A_SYNC and sample rates in low power mode 1

ACDC_CLK	D2A_CLK	D2A_SYNC	Sample rates supported
24.576MHz	24.576MHz	3.072MHz	12/24/48/96kHz
22.577MHz	22.577MHz	2.822MHz	11.024/22.05/44.1/88.2kHz
16.384MHz	16.384MHz	2.048MHz	8/16/32/64kHz

Table 4-6 Relationship of ACDC_CLK, D2A_CLK, D2A_SYNC and sample rates in low power mode 2

ACDC_CLK	D2A_CLK	D2A_SYNC	Sample rates supported
12.288MHz	1.536MHz	1.536MHz	12/24/48kHz
11.288MHz	1.411MHz	1.411MHz	11.024/22.05/44.1kHz
8.192MHz	1.024MHz	1.024MHz	8/16kHz

The Digital Audio Codec supports three application scenarios that are application mode 0, application mode 1 and application mode 2. Different application mode requires different ACDC_CLK.
For application mode 0 and application mode 1, there are no restrictions on ACDC_CLK. Just configure the frequency of ACDC_CLK according to sample rate and make sure that the frequency of ACDC_CLK is at least 8 times of D2A_SYNC. In this situation, the divider ACDCDIG_ADCINT_DIV and ACDCDIG_DACINT_DIV must be equal to 0x07.
For application mode 2, there are some differences. For example, sample rate of digital ADC is 12kHz while sample rate of DAC is 128kHz. Configure ACDC_CLK to the higher frequency such as 49.152MHz. The divider ACDCDIG_ADCINT_DIV must be set to 0x07 while ACDCDIG_DACINT_DIV must be set to 0x0b. On the contrary, if sample rate of digital ADC is 128kHz while sample rate of DAC is 12kHz. Configure ACDC_CLK to the higher frequency such as 49.152MHz. The divider ACDCDIG_ADCINT_DIV must be set to 8'h0b while ACDCDIG_DACINT_DIV must be set to 0x07.

4.6.2 Software Application Notes

Steps to configure Digital Audio Codec to start transfer are as follows.
1. Program CRU in the SOC system to achieve the frequency of ACDC_CLK.

2. Program ACDCDIG_ADCINT_DIV to 0x07 or 0x0b according to application modes.
3. Program ACDCDIG_ADCCLKCTRL to 0x3e. It is preferred to set cic_ds_ratio=2'b00 to make Digital Audio Codec work in normal mode.
4. Program ACDCDIG_ADCSCLKTXINT_DIV.
5. Program ACDCDIG_I2S_CKR0 and ACDCDIG_I2S_CKR1 to set I2S TX related registers.
6. Program ACDCDIG_DACINT_DIV to 0x07 or 0x0b according to application modes.
7. Program ACDCDIG_DACCLKCTRL to 0x3c.
8. Program ACDCDIG_DACSCLKRXINT_DIV.
9. Program ACDCDIG_I2S_CKR0 to set I2S RX related registers.
10. Program ACDCDIG_SYSCTRL0.clk_sel to select D2A_CLK source. Program ACDCDIG_SYSCTRL0.sync_sel to select D2A_SYNC source. Don't enable glbcke at this time.
11. Program I2C related registers to set properly device address, timeout value and so on. Then program ACDCDIG_I2C_CON0 to enable I2C at last.
12. Program ACDCDIG_I2S_CLR to clear I2S TX and RX logic.
13. Program ACDCDIG_I2S_TXCR0, ACDCDIG_I2S_TXCR1, ACDCDIG_I2S_RXCR0 and ACDCDIG_I2S_RXCR1.
14. Program I2S controller that is connected to Digital Audio Codec. Don't start I2S TX and RX at this time.
15. Program digital ADC related registers such as ACDCDIG_ADCHPFEN, ACDCDIG_ADCVUCTL, ACDCDIG_ADCCFG1 to achieve basic configuration.
16. Program DAC related registers such as ACDCDIG_DACHPF, ACDCDIG_DACVUCTL and ACDCDIG_DACCFG1 to achieve basic configuration.
17. Program ACDCDIG_I2S_XFER to start I2S TX and RX.
18. Program I2S controller outside Digital Audio Codec to start I2S TX and RX.
19. Program ACDCDIG_SYSCTRL0.glbcke to 1'b1 to enable output of D2A_SYNC.
20. Program ACDCDIG_ADCDIGEN and ACDCDIG_DACDIGEN to enable digital ADC channels and DAC channels. From now on, the Digital Audio Codec begins to work.

Steps to configure Digital Audio Codec to end transfer are as follows.

1. Program ACDCDIG_ADCDIGEN and ACDCDIG_DACDIGEN to disable digital ADC channels and DAC channels.
2. Program ACDCDIG_ADCCLKCTRL.adc_sync_ena and 1'b0 or ACDCDIG_DACCLKCTRL. dac_sync_ena to 1'b0 to disable the D2A_SYNC. Wait ACDCDIG_ADCCLKCTRL. adc_sync_status and ACDCDIG_DACCLKCTRL.dac_sync_status until both of them read back to be 1'b0.
3. Program ACDCDIG_SYSCTRL0.glbcke to 1'b0.

Chapter 5 Video Output Processor

5

5.1 Overview

VOP is the display interface from Direct Memory Access (DMA) to MCU-LCD. It's connected to an AHB bus. After configured by CPU, VOP is filled with pixel data transferred by DMA through AHB bus.

Features

- Bus interface
  - Support AMBA 2.0 AHB slave interface for accessing internal registers，32-bit data bus width
- Support source data format
  - RGB565
  - YUV420
  - Support UV swap
- Support YUV2RGB
  - Support BT601 limited range
  - Support BT709 limited range
  - Support BT601 full range
- Support allegro dither down for RGB888 to RGB565
- Support RGB565 display data format
  - Support display data swap
  - Half-word swap
  - Byte swap
- Support max output resolution 480x320
- Built-in i8080 MCU interface
  - 8-bit data bus width
  - Timing configurable
  - Data direction control when idle
  - Write split transfer mode support

5.2 Block Diagram

The VOP architecture is shown in figure 1-1. There are 4 data paths as shown below:

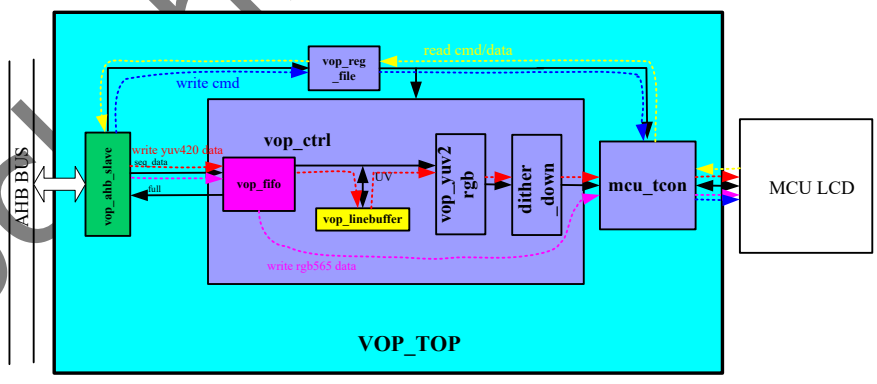

Fig. 5-1 VOP Block Diagram

1.RGB format through register and command path(blue dotted line): in order to be compatible with NanoC, RGB data can go through the registers directly.
2.RGB format through FIFO (pink dotted line): RGB data go through to MCU_TCON.
3.YUV format path (red dotted line): UV data go through from FIFO to LineBuffer, and will be fetched out at the time with Y data from FIFO, after translating from YUV to RGB888, dither down, data will be sended out.
4.Read path (yellow dotted line): RGB data or command read path

5.3 Function Description

5.3.1 Pixel format

1.RGB

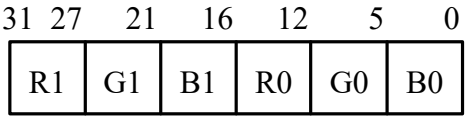

Fig. 5-2 RGB565 data format

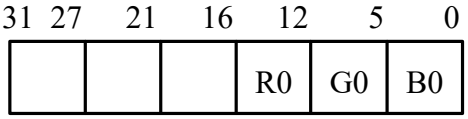

Fig. 5-3 RGB565 data format

2.YUV

V3	U3	V2	U2	V1	U1	V0	U0
Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
Y15	Y14	Y13	Y12	Y11	Y10	Y9	Y8
V7	U7	V6	U6	V5	U5	V4	U4

Fig. 5-4 YUV420 data format

U3	V3	U2	V2	U1	V1	U0	V0
Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
Y15	Y14	Y13	Y12	Y11	Y10	Y9	Y8
U7	V7	U6	V6	U5	V5	U4	V4

Fig. 5-5 YUV420 data format

5.3.2 Pixel Data Path

Pixel data is passed from memory to VOP through DMA as shown in figure1-4. It' required that YUV420 data form transferred from DMA is as shown in figure1-3 .

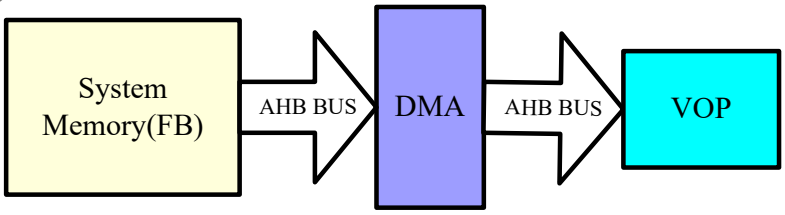

Fig. 5-6 pixel data path

5.3.3 Display process

1. Color space conversion

Support three standards for YUV2RGB as shown below .

yuv to rgb (REC-601) range 0 (Y[16:235], UV[16:240], RGB[0:255])

R = 1.164(Y-16) + 1.596(V-128)

G = 1.164(Y-16) - 0.391(U-128) - 0.813(V-128)

B = 1.164(Y-16) + 2.018(U-128)

yuv to rgb (REC-601) range 1 (YUV[0:255], RGB[0:255])
R = (Y-16) + 1.402(V-128)
G = (Y-16) - 0.344(U-128) - 0.714(V-128)
B = (Y-16) + 1.772(U-128)

yuv to rgb (REC-709) range 0 (Y[16:235], UV[16:240], RGB[0:255])
R = 1.164(Y-16) + 1.793(V-128)
G = 1.164(Y-16) - 0.213(U-128) - 0.534(V-128)
B = 1.164(Y-16) + 2.115(U-128)

2. Allegro Dither Down
Dithering is an intentional applied form of noise, using to randomize quantization error, and thereby preventing large-scaling patterns such as "banding".
The pixel value is used by dithering process to display the data in a lower color depth on the LCD panel . the source input format is RGB888 and display output format is RGB565. When dithering is enable (sw_mcu_dither_en) , the output data is generated by dithering algorithm based on the pixel position and the value of removed bits. Otherwise , the MSBs of the pixel color components are outputted as display data as shown in figure1-5.

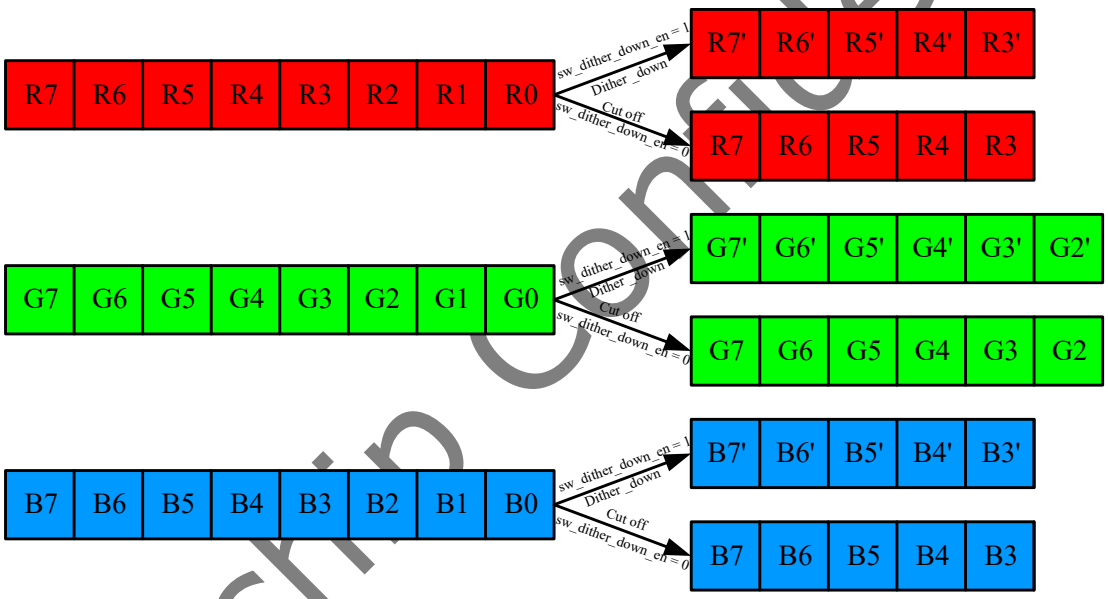

Fig. 5-7 dither down block

5.3.4 I/O description

VOP have 11 outputs connected to pad , which are controlled by IOMUX as shown in Table 1-1 .

Table 5-1 VOP output pins

Module pin	Dir	Pad name	IOMUX
lcd_d0	I/O	LCD_D0/CIF_D0/I2C0_SDA_M2/TKEY12/M4F_WFI/M4F_JTAG_TCK/M0_JTAG_TCK/AONJTAG_TCK/DSPJTAG_TCK/GPIO0_A0_u	GRF_GPIO0A_IOMUX_L[3:0]=4'h1
lcd_d1	I/O	LCD_D1/CIF_D1/I2C0_SCL_M2/TKEY13/M0_WFI/M4F_JTAG_TMS/M0_JTAG_TMS/AONJTAG_TMS/DSPJTAG_TMS/GPIO0_A1_u	GRF_GPIO0A_IOMUX_L[7:4]=4'h1
Lcd_rs	O	LCD_RS/CIF_HREF/I2C1_SDA_M2/TKEY14/TKEY_DRIVE_M3/PMU_STATE0/	GRF_GPIO0A_IOMUX_L[11:8]=4'h1

Module pin	Dir	Pad name	IOMUX
		AONJTAG_TDI/DSPJTAG_TDI/GPIO0_A2_u	
Lcd_csn	O	LCD_CSn/CIF_VSYNC/I2C1_SCL_M2/TKEY15/PMU_DEBUG/PMU_STATE1/AONJTAG_TDO/DSPJTAG_TDO/GPIO0_A3_u	GRF_GPIO0A_IOMUX_L[15:12]=4'h1
Lcd_rdn	O	LCD_RDN/CIF_CLKOUT/UART1_CTSN_M1/TKEY16/PMU_SLEEP/PMU_STATE2/CODEC_CLK_M1/AONJTAG_TRSTn/DSPJTAG_TRSTn/GPIO0_A4_u	GRF_GPIO0A_IOMUX_H[3:0]=4'h1
Lcd_wrn	O	LCD_WRN/CIF_CLKIN/UART1_RTSN_M1/PDM_CLK_M0/TKEY17/PMU_STATE3/CODEC_SYNC_M1/GPIO0_A5_d	GRF_GPIO0A_IOMUX_H[7:4]=4'h1
lcd_d2	I/O	LCD_D2/CIF_D2/UART1_RX_M1/PDM_SDI_M0/TKEY18/PMU_STATE4/CODEC_ADC_D_M1/GPIO0_A6_d	GRF_GPIO0A_IOMUX_H[11:8]=4'h1
lcd_d3	I/O	LCD_D3/CIF_D3/UART1_TX_M1/PDM_CLK_S_M0/TKEY19/TEST_CLKOUT/CODEC_DAC_DL_M1/GPIO0_A7_d	GRF_GPIO0A_IOMUX_H[15:12]=4'h1
lcd_d4	I/O	LCD_D4/CIF_D4/UART2_CTSN_M1/SPI1_CS0n_M1/SPI_SLV_CSn/GPIO0_B0_u	GRF_GPIO0B_IOMUX_L[3:0]=4'h1
lcd_d5	I/O	LCD_D5/CIF_D5/UART2_RTSN_M1/SPI1_CLK_M1/SPI_SLV_CLK/GPIO0_B1_u	GRF_GPIO0B_IOMUX_L[7:4]=4'h1
lcd_d6	I/O	LCD_D6/CIF_D6/UART2_RX_M1/SPI1_MOSI_M1/SPI_SLV_MOSI/GPIO0_B2_u	GRF_GPIO0B_IOMUX_L[11:8]=4'h1
lcd_d7	I/O	LCD_D7/CIF_D7/UART2_TX_M1/SPI1_MISO_M1/SPI_SLV_MISO/GPIO0_B3_u	GRF_GPIO0B_IOMUX_L[15:12]=4'h1

5.4 Register Description

5.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
vop_mcu_con	0x0000	W	0x00000000	Mode control register
vop_mcu_version	0x0004	W	0x00000000	VOP version
vop_mcu_timing	0x0008	W	0x00000000	MCU interface timing control
vop_mcu_lcd_size	0x000c	W	0x00000000	Configure LCD size
vop_mcu_fifo_watermark	0x0010	W	0x00000000	Configure FIFO watermark
vop_mcu_srt	0x0014	W	0x00000000	VOP soft_reset
vop_mcu_int_en	0x0018	W	0x00000000	VOP interrupt enable
vop_mcu_int_clear	0x001c	W	0x00000000	VOP interrupt clear
vop_mcu_int_status	0x0020	W	0x00000000	VOP interrupt status

Name	Offset	Size	Reset Value	Description
vop_mcu_status	0x0024	W	0x00000000	VOP status register
vop_mcu_cmd	0x0028	W	0x00000000	VOP command r/w entry
vop_mcu_data	0x002c	W	0x00000000	VOP data r/w entry
vop_mcu_start	0x0030	W	0x00000000	start vop

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

5.4.2 Detail Register Description

vop_mcu_con

Address: Operational Base + offset (0x0000)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:13	RW	0x0	sw_mcu_burst mask dummy fifo writing for burst stride 3'b000 : SINGLE 3'b001 : INCR4 3'b010 : INCR8 3'b111 : INCR16
12	RW	0x0	sw_auto_ckg auto clock gating 1'b0: disable 1'b1: enable
11	RW	0x0	sw_wdata_bypass_en 1'b0: no bypass 1'b1: bypass
10	RW	0x0	sw_dither_down_en dither down enable 1'b0: disable 1'b1: enable
9	RW	0x0	sw_mcu_uv_swap write yuv420 data uv swap
8	RW	0x0	sw_mcu_input_format 1'b0: rgb565 1'b1: yuv420
7:6	RW	0x0	sw_mcu_y2r_mode yuv2rgb mode selection 2'b00 : BT601 limited 2'b01 : BT709 limited 2'b10 : BT601 fulled 2'b11 : reserved
5	RW	0x0	sw_mcu_byte_swap mcu write data half-word swap
4	RW	0x0	sw_mcu_hw_swap mcu write data half-word swap

Bit	Attr	Reset Value	Description
3	RW	0x0	sw_mcu_bits mcu data width 1'b0: 8bits 1'b1: 16bits
2:1	RW	0x0	sw_mcu_wr_phase write data split 2'b00 : no split 2'b01 : two phase split 2'b10 : three phase split 2'b11 : four phase split
0	RW	0x0	sw_mcu_idle_dir data out valid , when there is no write/read operation, data pin(D0 ~ D15) as input or output 1'b0: D0 ~ D15 input 1'b1: D0 ~ D15 output valid

vop_mcu_version

Address: Operational Base + offset (0x0004)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	sw_mcu_version Indicate VOP version

vop_mcu_timing

Address: Operational Base + offset (0x0008)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:12	RW	0x0	sw_mcu_csrw Chip select to R/W strobe leading edge this field specifies the number of processor clock cycles from the falling edge of CSN to the falling edge of RDN or WRN . If this bit is set to 0x0 , the processor uses a csrw valus of 0x1
11	RO	0x0	reserved
10:5	RW	0x00	sw_mcu_rwpw this field controls the width of RDN or WRN in processor clock cycles . If this bit is set to 0x0 , the processor uses an rwpw value of 0x1
4:0	RW	0x00	sw_mcu_rwcs this field specifies the number of processor clock cycles from the rising edge of RDN or WRN to the rising edge of CSN . If this is set to 0x0 , the processor uses an rwcs valus of 0x1

vop_mcu_lcd_size

Address: Operational Base + offset (0x000c)

Bit	Attr	Reset Value	Description
31:21	RO	0x0	reserved
20:12	RW	0x000	sw_mcu_lcd_height LCD height , max value is 400 , real -1
11:9	RO	0x0	reserved
8:0	RW	0x000	sw_mcu_lcd_width LCD width , max value is 400 , real -1

vop_mcu_fifo_watermark

Address: Operational Base + offset (0x0010)

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12:8	RW	0x00	sw_almost_empty_watermark Recommended value is not greater than 16
7:5	RO	0x0	reserved
4:0	RW	0x00	sw_almost_full_watermark Recommended value is not greater than 16 , and if it's set to a value less than 2 , real valus is 2

vop_mcu_srt

Address: Operational Base + offset (0x0014)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	W1C	0x0	sw_soft_reset auto clear 1'b0 : not clear 1'b1 : clear

vop_mcu_int_en

Address: Operational Base + offset (0x0018)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	sw_int_en_fifo_full 1'b0 : fifo_full interrupt disable 1'b1 : fifo_full interrupt enable
1	RW	0x0	sw_int_en_fifo_empty 1'b0 : fifo_empty interrupt disable 1'b1 : fifo_empty interrupt enable
0	RW	0x0	sw_int_en_frame_done 1'b0 : frame_done interrupt disable 1'b1 : frame_done interrupt enable

vop_mcu_int_clear

Address: Operational Base + offset (0x001c)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	sw_int_clear_fifo_full 1'b0 : no clear 1'b1 : clear
1	RW	0x0	sw_int_clear_fifo_empty 1'b0 : no clear 1'b1 : clear
0	RW	0x0	sw_int_clear_frame_done 1'b0 : no clear 1'b1 : clear

vop_mcu_int_status

Address: Operational Base + offset (0x0020)

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5	RW	0x0	sw_int_raw_fifo_full 1'b0: invalid 1'b1: valid
4	RW	0x0	sw_int_raw_fifo_empty 1'b0: invalid 1'b1: valid
3	RW	0x0	sw_int_raw_frame_done 1'b0: invalid 1'b1: valid
2	RW	0x0	sw_int_fifo_full 1'b0: invalid 1'b1: valid
1	RW	0x0	sw_int_fifo_empty 1'b0: invalid 1'b1: valid
0	RW	0x0	sw_int_frame_done 1'b0: invalid 1'b1: valid

vop_mcu_status

Address: Operational Base + offset (0x0024)

Bit	Attr	Reset Value	Description
31:25	RO	0x0	reserved
24:16	RW	0x000	sw_mcu_current_line mcu current line
15:13	RO	0x0	reserved
12:4	RW	0x000	sw_mcu_current_row mcu current row
3:1	RO	0x0	reserved

Bit	Attr	Reset Value	Description
0	RW	0x0	sw_mcu_working 1'b0 : idle 1'b1 : working

vop_mcu_cmd

Address: Operational Base + offset (0x0028)

Bit	Attr	Reset Value	Description
31:8	RW	0x000000	mcu_cmd_high_bits VOP command write (only in split mode)
7:0	RW	0x00	mcu_cmd_low_bits VOP command write VOP status read (only use low 8-bits)

vop_mcu_data

Address: Operational Base + offset (0x002c)

Bit	Attr	Reset Value	Description
31:8	RW	0x000000	mcu_data_high_bits VOP data write(only in split mode)
7:0	RW	0x00	mcu_data_low_bits VOP data write VOP data read (only use low 8-bit)

vop_mcu_start

Address: Operational Base + offset (0x0030)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	sw_mcu_start start vop 1'b0 : no valid 1'b1 : valid

5.5 Timing Diagram

5.5.1 i8080 Timing

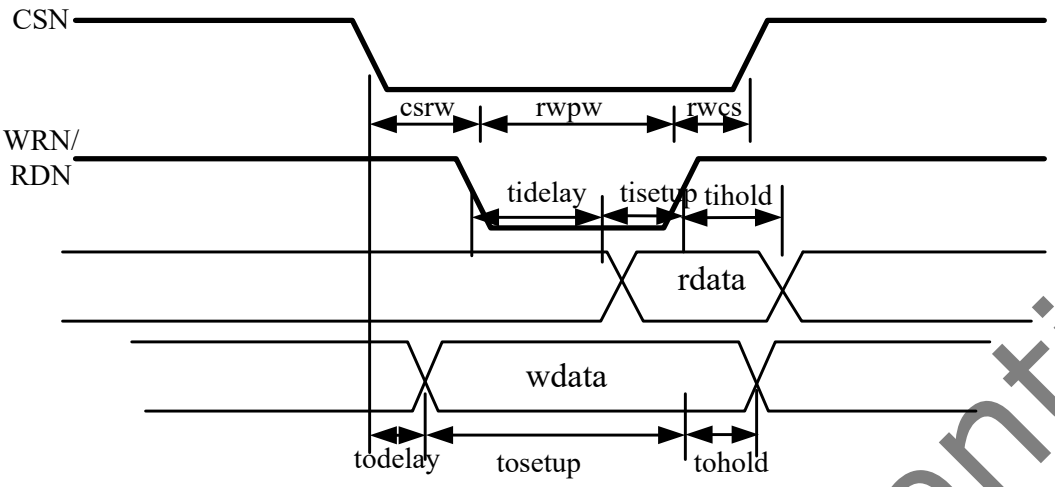

Fig. 5-8 i8080 r/w timing

5.5.2 write data split

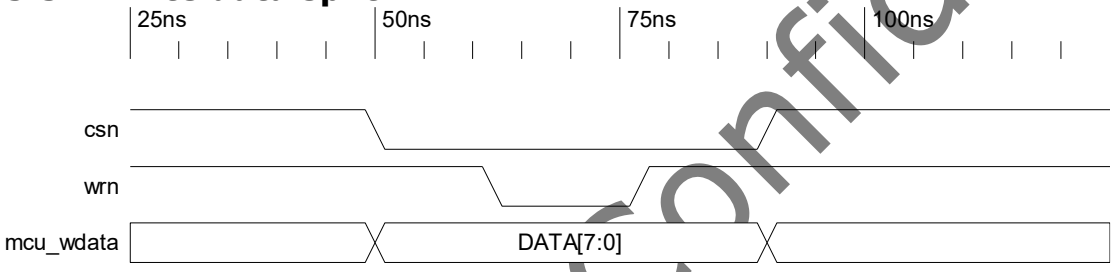

Fig. 5-9 no split

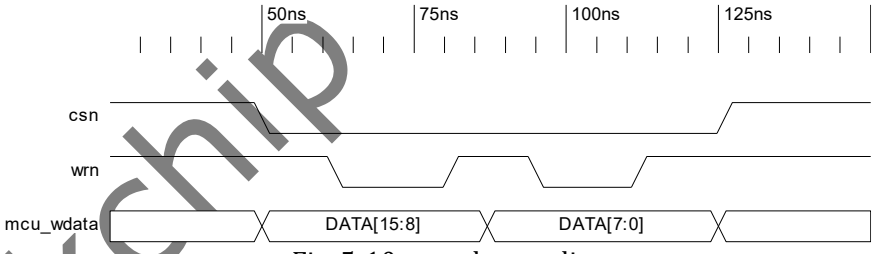

Fig. 5-10 two-phase split

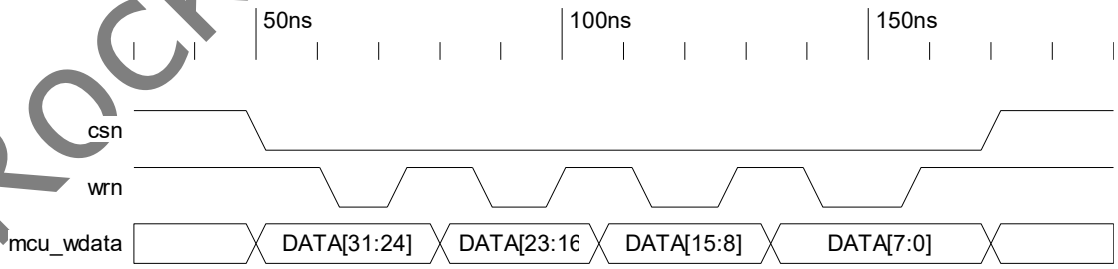

Fig. 5-11 four-phase split

5.6 Application Notes

5.6.1 Configure registers

The flow of configuring register is shown in figure 4-1.

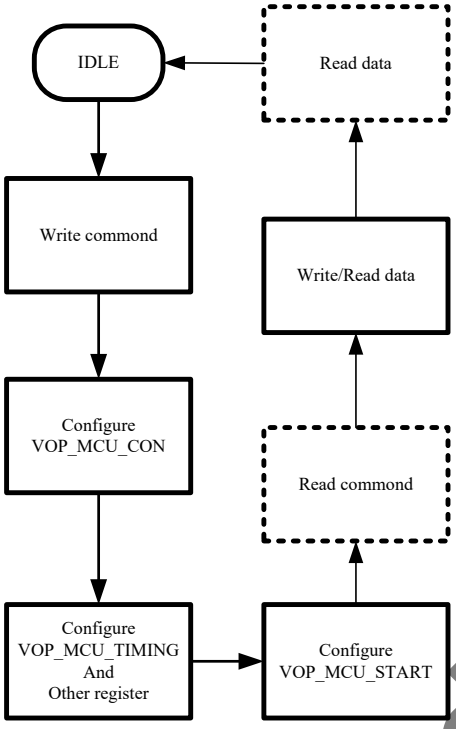

Fig. 5-12 operation flow

Note1: Registers can be configured again until VOP is idle when sw_mcu_working is 1'b0 .
Note2: Can't read data/command or write command until VOP is idle.
Note3: Before a new picture is sent , vop_mcu_start must be configured again .

5.6.2 Input RGB565 data

Note1: If the source data format is RGB565, set sw_mcu_input_format to 0 . Otherwise, sw_mcu_input_format is set to 1.
Note2：If the RGB565 source data is stored as figure1-2a，sw_mcu_hw_swap must be set to 1'b1 and sw_mcu_wr_phase must be set to 2'b11.
Note3：If the RGB565 source data is stored as figure1-2b，sw_mcu_hw_swap must be set to 1'b0 and sw_mcu_wr_phase must be set to 2'b01.

5.6.3 Input YUV420 data

Note1: If sw_dither_down_en is not asserted , the LBS of output data will be cut off directly .
Note2: If the YUV420 source data is stored as figure1-3b, sw_mcu_uv_swap must be asserted .
Note3: If the source data format is YUV420, sw_mcu_wr_phase must be set 2'b01.

5.6.4 Read commond or data

Note1：Set sw_mcu_uv_swap to be 2'b00 to read commond or data.

Chapter 6 Video Capture(VICAP)

6.1 Overview

The Video Capture, receives the data from Camera via DVP, and transfers the data into system main memory by AXI bus.
The features of VICAP are as follow:

- Support BT601 YCbCr 422 8bit input
- Support BT656 YCbCr 422 8bit input
- Support UYVY/VYUY/YUYV/YVYU configurable
- Support RAW 8/10/12 bit input
- Support JPEG input
- Support window cropping
- Support virtual stride when write to memory
- Support different stored address for Y and UV
- Support 422/420/400 output
- Support one frame/frame ping-pong/block ping-pong mode
- Support the polarity of pixel_clk,hsync,vsync configurable

6.2 Block Diagram

VICAP comprises with:

- AHB Slave

Host configure the registers via the AHB Slave

- AXI Master

Transmit the data to chip memory via the AXI Master

- INTERFACE

Translate the input video data into the requisite data format

- CROP

Bypass or crop the source video data to a smaller size destination

- DMA

Control the operation of AXI Master

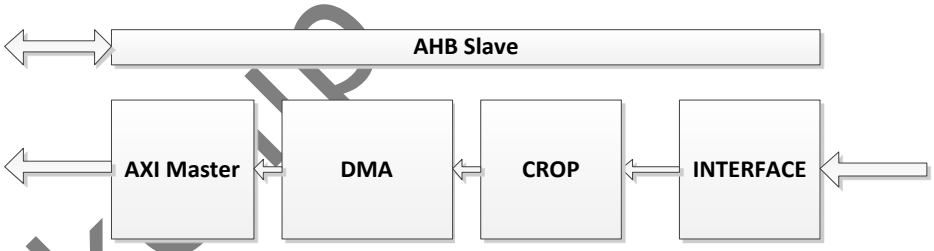

Fig. 6-1 VICAP Block Diagram

6.3 Function Description

This chapter is used to illustrate the operational behavior of how VICAP works. If YUV422 or ccir656 signal is received from external devices, VICAP translate it into YUV422/420 data, and separate the data to Y and UV data, then store them to different memory via AXI bus separately.

6.3.1 Support Vsync high active or low active

- Vsync Low active as below

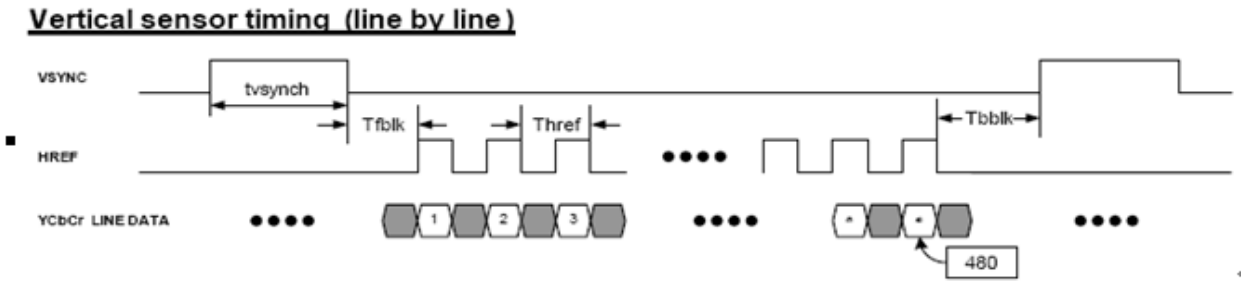

Fig. 6-2 Timing diagram for VICAP when vsync low active

- Vsync High active

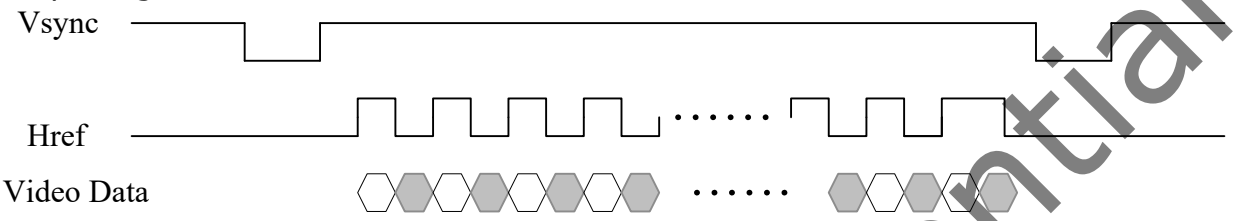

Fig. 6-3 Timing diagram for VICAP when vsync high active

6.3.2 Support href high active or low active

- Href high active

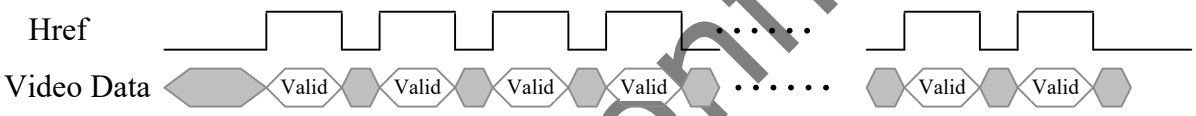

Fig. 6-4 Timing diagram for VICAP when href high active

- Href Low active

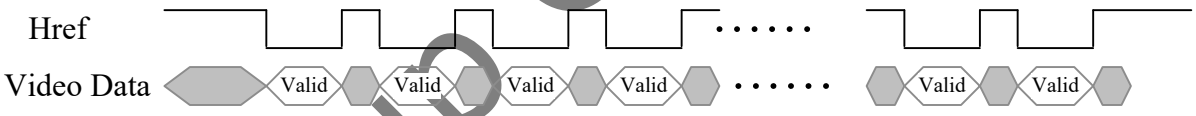

Fig. 6-5 Timing diagram for VICAP when href low active

- Y first

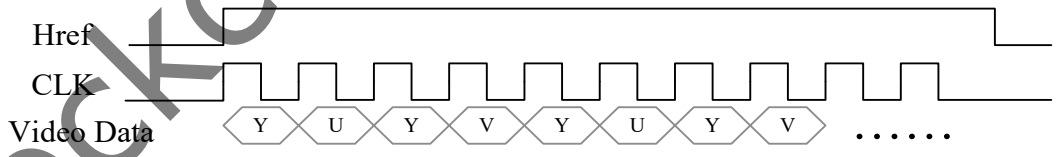

Fig. 6-6 Timing diagram for VICAP when Y data first

- U first

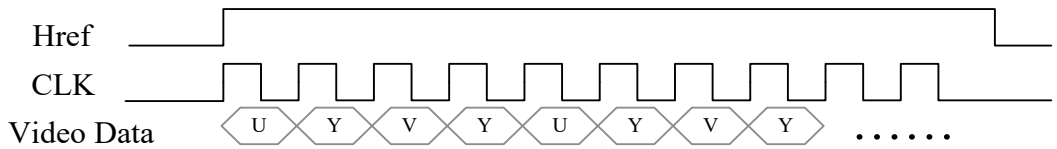

Fig. 6-7 Timing diagram for VICAP when U data first

6.3.3 Support CCIR656 (NTSC and PAL)

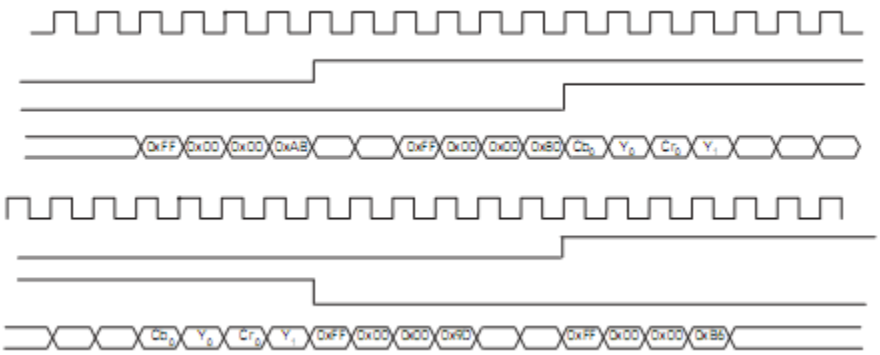

Fig. 6-8 CCIR656 timing

6.3.4 Support Raw data (8-bit) or JPEG

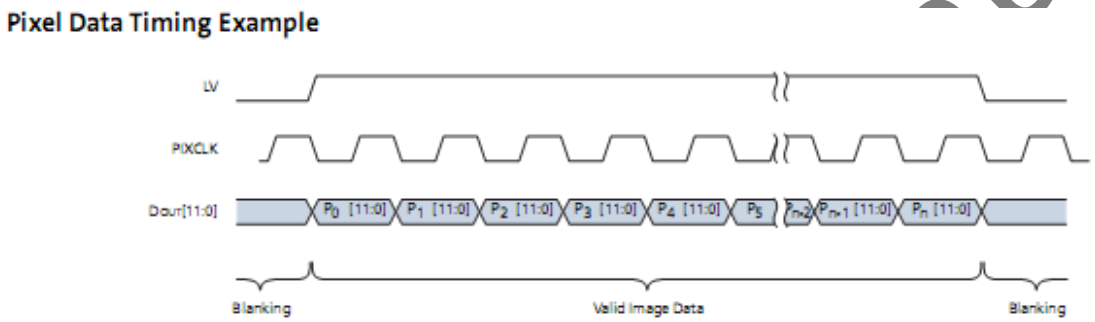

Fig. 6-9 Raw Data or JPEG Timing

VICAP module can work in three modes: one frame stop mode, frame ping-pong mode, frame block mode.

One frame stop mode

In this mode, configure the parameter WORK_MODE to one frame stop mode. After one frame captured, VICAP will automatic stop and set the status of FRAME_STATUS. VICAP will not capture the sensor data until the user clear the FRAME_STATUS. The address of Y/UV is FRAME0_ADDR or FRAME1_ADDR in turn.

Frame Ping-Pong mode

After one frame(F1) captured, VICAP will start to capture the next frame(F2) automatically, and host must assign new address pointer of frame1 and clear the frame1 status, thus VICAP will capture the third frame automatically(by new F1 address) without any stop and so on for the following frames. But if host did not update the frame buffer address, the VICAP will cover the pre-frame data stored in the memory with the following frame data.

Block Ping-Pong mode

In this mode, VICAP will work in a unit of block. The number of lines of block is decided by the configuration of BLOCK_LINE_NUM. Block0 and block1 are received in turn. When block0/1 is done, the BLOCK_STATUS will be set, and the user should clear the BLOCK_STATUS in time. When the next block0/1 is beginning to be received, if the BLOCK_STATUS_0/1 is not cleared, the rest of the current frame will be discarded.

Storage

Difference between the YUV mode and RAW mode is that in the YUV mode or CCIR656 mode, data will be storage in the Y data buffer and UV data buffer,and if the only y mode is choosed the UV data will not be storage; in the RAW or JPEG mode, RGB data will be storage in the same buffer. In addition, in the YUV mode or RAW8 mode, the width of Y, U or V data is a byte in memory; in Raw10/12 or JPEG mode, the width is a halfword.

CROP

The parameter START_Y and START_X defines the coordinate of crop start point. And the frame size after cropping is following the value of SET_WIDTH and SET_HEIGHT.

6.4 Register Description

6.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
VICAP_DVP_CTRL	0x0000	W	0x00007000	DVP path control
VICAP_DVP_INTEN	0x0004	W	0x00000000	DVP path interrupt status
VICAP_DVP_INTSTAT	0x0008	W	0x00000000	DVP path interrupt status
VICAP_DVP_FOR	0x000c	W	0x00000000	DVP path format
VICAP_DVP_DMA_IDLE_REQ	0x0010	W	0x00000000	DVP path dma module idle requeset
VICAP_DVP_FRM0_ADDR_Y	0x0014	W	0x00000000	DVP path frame0 y address
VICAP_DVP_FRM0_ADDR_UV	0x0018	W	0x00000000	DVP path frame0 uv address
VICAP_DVP_FRM1_ADDR_Y	0x001c	W	0x00000000	DVP path frame1 y address
VICAP_DVP_FRM1_ADDR_UV	0x0020	W	0x00000000	DVP path frame1 uv address
VICAP_DVP_VIR_LINE_WIDTH	0x0024	W	0x00000000	DVP path virtual line width
VICAP_DVP_SET_SIZE	0x0028	W	0x01e002d0	The expected width and height of received image
VICAP_DVP_BLOCK_LINE_NUM	0x002c	W	0x00000000	The line number of block
VICAP_DVP_BLOCK0_ADDR_Y	0x0030	W	0x00000000	DVP path block0 y address
VICAP_DVP_BLOCK0_ADDR_UV	0x0034	W	0x00000000	DVP path block0 uv address
VICAP_DVP_BLOCK1_ADDR_Y	0x0038	W	0x00000000	DVP path block1 y address
VICAP_DVP_BLOCK1_ADDR_UV	0x003c	W	0x00000000	DVP path block1 uv address
VICAP_DVP_BLOCK_STATUS	0x0040	W	0x00000003	The status of block
VICAP_DVP_CROP	0x0044	W	0x00000000	The start point of DVP path cropping
VICAP_DVP_PATH_SEL	0x0048	W	0x00000000	DVP path selection
VICAP_DVP_LINE_INT_NUM	0x004c	W	0x00000000	DVP path line number of line interrupt
VICAP_DVP_WATER_LINE	0x0050	W	0x00000000	DVP path water line
VICAP_DVP_FIFO_ENTRY	0x0054	W	0x00000000	DVP path FIFO entry
VICAP_DVP_FRAME_STATUS	0x0060	W	0x00000000	DVP path frame status

Name	Offset	Size	Reset Value	Description
VICAP_DVP_CUR_DST	0x0064	W	0x00000000	DVP path current destination address
VICAP_DVP_LAST_LINE	0x0068	W	0x00000000	DVP path last frame line number
VICAP_DVP_LAST_PIX	0x006c	W	0x00000000	DVP path last line pixel number

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

6.4.2 Detail Register Description

VICAP_DVP_CTRL

Address: Operational Base + offset (0x0000)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:12	RW	0x7	axi_burst_type 0-15 : Burst1~16
11:3	RO	0x0	reserved
2:1	RW	0x0	work_mode 2'b00: One frame stop mode 2'b01: Frame ping-pong mode 2'b02: Block ping-pong mode 2'b03: Reserved
0	RW	0x0	cap_en 1'b0: Disable 1'b1: Enable

VICAP_DVP_INTEN

Address: Operational Base + offset (0x0004)

Bit	Attr	Reset Value	Description
31:15	RO	0x0	reserved
14	RW	0x0	block_err_en Block error interrupt enable 1'b0: Disable 1'b1: Enable
13	RW	0x0	line1_end_en Line1 end interrupt enable 1'b0: Disable 1'b1: Enable
12	RW	0x0	line0_end_en Line0 end interrupt enable 1'b0: Disable 1'b1: Enable
11	RW	0x0	block1_end_en Block1 end interrupt enable 1'b0: Disable 1'b1: Enable

Bit	Attr	Reset Value	Description
10	RW	0x0	block0_end_en Block0 end interrupt enable 1'b0: Disable 1'b1: Enable
9	RW	0x0	pst_inf_frame_end_en Frame end after interface FIFO interrupt enable 1'b0: Disable 1'b1: Enable
8	RW	0x0	pre_inf_frame_end_en Frame end before interface FIFO interrupt enable 1'b0: Disable 1'b1: Enable
7	RW	0x0	frame_start_en Interface frame start interrupt enable 1'b0: Disable 1'b1: Enable
6	W1C	0x0	bus_err_en Axi master or ahb slave response error interrupt enable 1'b0: Disable 1'b1: Enable
5	RW	0x0	dfifo_of_en DMA FIFO overflow interrupt enable 1'b0: Disable 1'b1: Enable
4	RW	0x0	ififo_of_en Interface FIFO overflow interrupt enable 1'b0: Disable 1'b1: Enable
3	W1C	0x0	pix_err_en The pixel number of last line not equal to the set height interrupt enable 1'b0: Disable 1'b1: Enable
2	W1C	0x0	line_err_en The line number of last frame not equal to the set height interrupt enable 1'b0: Disable 1'b1: Enable
1	W1C	0x0	line_end_en Line end interrupt enable 1'b0: Disable 1'b1: Enable

Bit	Attr	Reset Value	Description
0	W1C	0x0	dma_frame_end_en Dma frame end interrupt enable 1'b0: Disable 1'b1: Enable

VICAP_DVP_INTSTAT

Address: Operational Base + offset (0x0008)

Bit	Attr	Reset Value	Description
31:15	RO	0x0	reserved
14	RW	0x0	block_err Block error end interrupt 1'b0: No interrupt 1'b1: Interrupt
13	RW	0x0	line1_end Dma line1 end interrupt 1'b0: No interrupt 1'b1: Interrupt
12	RW	0x0	line0_end Dma line0 end interrupt 1'b0: No interrupt 1'b1: Interrupt
11	RW	0x0	block1_end Dma block1 end interrupt 1'b0: No interrupt 1'b1: Interrupt
10	RW	0x0	block0_end Dma block0 end interrupt 1'b0: No interrupt 1'b1: Interrupt
9	RW	0x0	pst_inf_frame_end Frame end after interface FIFO interrupt 1'b0: No interrupt 1'b1: Interrupt
8	RW	0x0	pre_inf_frame_end Frame end before interface FIFO interrupt 1'b0: No interrupt 1'b1: Interrupt
7	RW	0x0	frame_start Interface frame start interrupt 1'b0: No interrupt 1'b1: Interrupt
6	W1C	0x0	bus_err Axi master or ahb slave response error interrupt 1'b0: No interrupt 1'b1: Interrupt

Bit	Attr	Reset Value	Description
5	RW	0x0	dfifo_of DMA FIFO overflow interrupt 1'b0: No interrupt 1'b1: Interrupt
4	RW	0x0	ififo_of Interface FIFO overflow interrupt 1'b0: No interrupt 1'b1: Interrupt
3	W1C	0x0	pix_err The pixel number of last line not equal to the set height interrupt 1'b0: No interrupt 1'b1: Interrupt
2	W1C	0x0	line_err The line number of last frame not equal to the set height interrupt 1'b0: No interrupt 1'b1: Interrupt
1	W1C	0x0	line_end Line end interrupt 1'b0: No interrupt 1'b1: Interrupt
0	W1C	0x0	dma_frame_end Dma frame end interrupt 1'b0: No interrupt 1'b1: Interrupt

VICAP_DVP_FOR

Address: Operational Base + offset (0x000c)

Bit	Attr	Reset Value	Description
31:21	RO	0x0	reserved
20	RW	0x0	hsync_mode 1'b0:normal mode(have nothing to do with vsync) 1'b1:vsync valid mode
19	RW	0x0	uv_store_order 1'b0: UVUV 1'b1: VUVU
18	RW	0x0	raw_end 1'b0: Little end 1'b1: Big end
17	RW	0x0	out_420_order 1'b0: UV in the even line 1'b1: UV in the odd line Note: The first line is even line(line 0)

Bit	Attr	Reset Value	Description
16	RW	0x0	output_420 1'b0: Output is 422 1'b1: Output is 420
15	RW	0x0	only_y_mode only receive y data for BT601
14:13	RO	0x0	reserved
12:11	RW	0x0	raw_width 2'b00: 8bit raw data; 2'b01: 10bit raw data; 2'b10: 12bit raw data; 2'b11: Reserve;
10	RW	0x0	jpeg_mode 1'b0: Other mode 1'b1: Mode1
9	RW	0x0	field_order 1'b0: Odd field first 1'b1: Even field first
8:7	RO	0x0	reserved
6:5	RW	0x0	yuv_in_order 2'b00: UYVY 2'b01: YVYU 2'b10: VYUY 2'b11: YUYV
4:2	RW	0x0	input_mode 3'b000: YUV 3'b010: PAL 3'b011: NTSC 3'b100: RAW 3'b101: JPEG 3'b110: Reserved 3'b111: Reserved
1	RW	0x0	href_pol 1'b0: High active 1'b1: Low active
0	RW	0x0	vsync_pol 1'b0: Low active 1'b1: High active

VICAP_DVP_DMA_IDLE_REQ

Address: Operational Base + offset (0x0010)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	dma_idle_req Write 1 will stop VICAP dma. When this bit change to 0,dma is stopped really

VICAP_DVP_FRM0_ADDR_Y

Address: Operational Base + offset (0x0014)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	frm0_addr_y DVP path frame0 y address

VICAP_DVP_FRM0_ADDR_UV

Address: Operational Base + offset (0x0018)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	frm0_addr_uv DVP path frame0 uv address

VICAP_DVP_FRM1_ADDR_Y

Address: Operational Base + offset (0x001c)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	frm1_addr_y DVP path frame1 y address

VICAP_DVP_FRM1_ADDR_UV

Address: Operational Base + offset (0x0020)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	frm1_addr_uv DVP path frame1 uv address

VICAP_DVP_VIR_LINE_WIDTH

Address: Operational Base + offset (0x0024)

Bit	Attr	Reset Value	Description
31:15	RO	0x0	reserved
14:0	RW	0x0000	vir_line_width DVP path virtual line width

VICAP_DVP_SET_SIZE

Address: Operational Base + offset (0x0028)

Bit	Attr	Reset Value	Description
31:29	RO	0x0	reserved
28:16	RW	0x01e0	set_height The expected height of received image
15:13	RO	0x0	reserved
12:0	RW	0x02d0	set_width The expected width of received image

VICAP_DVP_BLOCK_LINE_NUM

Address: Operational Base + offset (0x002c)

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12:0	RW	0x0000	block_line_num The line number of block.

VICAP_DVP_BLOCK0_ADDR_Y

Address: Operational Base + offset (0x0030)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	block0_addr_y DVP path block0 y address

VICAP_DVP_BLOCK0_ADDR_UV

Address: Operational Base + offset (0x0034)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	block0_addr_uv DVP path block0 uv address

VICAP_DVP_BLOCK1_ADDR_Y

Address: Operational Base + offset (0x0038)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	block1_addr_y DVP path block1 y address

VICAP_DVP_BLOCK1_ADDR_UV

Address: Operational Base + offset (0x003c)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	block1_addr_uv DVP path block1 uv address

VICAP_DVP_BLOCK_STATUS

Address: Operational Base + offset (0x0040)

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RO	0x00	blk_id The id of blocks in one frame. 0 represent for the first block
15:2	RO	0x0	reserved
1	W1C	0x1	blk1_status When blk1 is received completely,the blk1_status will be pulled up to 1,and blk1_status should be cleared before the start of blk1 of next frame

Bit	Attr	Reset Value	Description
0	W1C	0x1	blk0_status When blk0 is received completely,the blk0_status will be pulled up to 1,and blk0_status should be cleared before the start of blk0 of next frame

VICAP_DVP_CROP

Address: Operational Base + offset (0x0044)

Bit	Attr	Reset Value	Description
31:29	RO	0x0	reserved
28:16	RW	0x0000	start_y The vertical ordinate of the start point
15:13	RO	0x0	reserved
12:0	RW	0x0000	start_x The horizontal ordinate of the start point

VICAP_DVP_PATH_SEL

Address: Operational Base + offset (0x0048)

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5	RW	0x0	raw_sel 1'b0: Not select 1'b1: Select
4	RW	0x0	yuv_sel 1'b0: Not select 1'b1: Select
3:0	RO	0x0	reserved

VICAP_DVP_LINE_INT_NUM

Address: Operational Base + offset (0x004c)

Bit	Attr	Reset Value	Description
31:29	RO	0x0	reserved
28:16	RW	0x0000	line1_int_num Line number
15:13	RO	0x0	reserved
12:0	RW	0x0000	line0_int_num Line number

VICAP_DVP_WATER_LINE

Address: Operational Base + offset (0x0050)

Bit	Attr	Reset Value	Description
31:10	RO	0x0	reserved
9:8	RW	0x0	water_line Water line of DMA fifo for hurry generation 2'b00: 75% 2'b01: 50% 2'b10: 25% 2'b11: 0%
7:6	RO	0x0	reserved
5:4	RW	0x0	hurry_value Value of hurry
3:1	RO	0x0	reserved
0	RW	0x0	hurry_en DMA hurry enable 1'b0: disable 1'b1: enable

VICAP_DVP_FIFO_ENTRY

Address: Operational Base + offset (0x0054)

Bit	Attr	Reset Value	Description
31:18	RO	0x0	reserved
17:9	RW	0x000	uv_fifo_entry Write 0 clear
8:0	RO	0x000	y_fifo_entry Write 0 clear

VICAP_DVP_FRAME_STATUS

Address: Operational Base + offset (0x0060)

Bit	Attr	Reset Value	Description
31:16	RO	0x0000	frame_num Completed frame number Write 0 to clear
15:3	RO	0x0	reserved
2	RO	0x0	idle 1'b0: Work 1'b1: Idle
1	RO	0x0	f1_sts 1'b0: Frame 1 not ready 1'b1: Frame 1 ready Write 0 clear
0	RO	0x0	f0_sts 1'b0: Frame 0 not ready 1'b1: Frame 0 ready Write 0 clear

VICAP_DVP_CUR_DST

Address: Operational Base + offset (0x0064)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	cur_dst DVP path current destination address

VICAP_DVP_LAST_LINE

Address: Operational Base + offset (0x0068)

Bit	Attr	Reset Value	Description
31:29	RO	0x0	reserved
28:16	RW	0x0000	last_uv_num UV line number of last frame,only for bt1120 mode
15:13	RO	0x0	reserved
12:0	RO	0x0000	last_y_num Y line number of last frame

VICAP_DVP_LAST_PIX

Address: Operational Base + offset (0x006c)

Bit	Attr	Reset Value	Description
31:29	RO	0x0	reserved
28:16	RW	0x0000	last_uv_num DVP path last line uv number
15:13	RO	0x0	reserved
12:0	RO	0x0000	last_y_num DVP path last line y number

6.5 Interface Description

Table 6-1 VICAP Interface Description

Module Pin	Direction	Pin Name	IOMUX Setting
vip_clkout	O	LCD_RDN/CIF_CLKOUT/UART1_CTSN_M1/TKEY16/PMU_SLEEP/PMU_STATE2/CODEC_CLK_M1/AONJTAG_TRSTn/DSPJTAG_TRSTn/GPIO0_A4_u	GRF_GPIO0A_IOMUX_SEL_H[3:0] = 4'h2
vip_clkin	I	LCD_WRN/CIF_CLKIN/UART1_RTSN_M1/PDM_CLK_M0/TKEY17/PMU_STATE3/CODEC_SYNC_M1/GPIO0_A5_d	GRF_GPIO0A_IOMUX_SEL_H[7:4] = 4'h2
vip_href	I	LCD_RS/CIF_HREF/I2C1_SDA_M2/TKEY14/TKEY_DRIVE_M3/PMU_STATE0/AONJTAG_TDI/DSPJTAG_TDI/GPIO0_A2_u	GRF_GPIO0A_IOMUX_SEL_L[11:8] = 4'h2
vip_vsync	I	LCD_CSn/CIF_VSYNC/I2C1_SCL_M2/TKEY15/PMU_DEBUG/PMU_STATE1/AONJTAG_TDO/DSPJTAG_TDO/GPIO0_A3_u	GRF_GPIO0A_IOMUX_SEL_L[15:12] = 4'h2

Module Pin	Direction	Pin Name	IOMUX Setting
vip_data0	I	LCD_D0/CIF_D0/I2C0_SDA_M2/TKEY12/M4F_WFI/M4F_JTAG_TCK/M0_JTAG_TCK/AONJTAG_TCK/DSPJTAG_TCK/GPIO0_A0_u	GRF_GPIO0A_IOMUX_SEL_L[3:0] = 4'h2
vip_data1	I	LCD_D1/CIF_D1/I2C0_SCL_M2/TKEY13/M0_WFI/M4F_JTAG_TMS/M0_JTAG_TMS/AONJTAG_TMS/DSPJTAG_TMS/GPIO0_A1_u	GRF_GPIO0A_IOMUX_SEL_L[7:4] = 4'h2
vip_data2	I	LCD_D2/CIF_D2/UART1_RX_M1/PDM_SDI_M0/TKEY18/PMU_STATE4/CODEC_ADC_D_M1/GPIO0_A6_d	GRF_GPIO0A_IOMUX_SEL_H[11:8] = 4'h2
vip_data3	I	LCD_D3/CIF_D3/UART1_TX_M1/PDM_CLK_S_M0/TKEY19/TEST_CLKOUT/CODEC_DAC_DL_M1/GPIO0_A7_d	GRF_GPIO0A_IOMUX_SEL_H[15:12] = 4'h2
vip_data4	I	LCD_D4/CIF_D4/UART2_CTSN_M1/SPI1_CS0n_M1/SPI_SLV_CSn/GPIO0_B0_u	GRF_GPIO0B_IOMUX_SEL_L[3:0] = 4'h2
vip_data5	I	LCD_D5/CIF_D5/UART2_RTSN_M1/SPI1_CLK_M1/SPI_SLV_CLK/GPIO0_B1_u	GRF_GPIO0B_IOMUX_SEL_L[7:4] = 4'h2
vip_data6	I	LCD_D6/CIF_D6/UART2_RX_M1/SPI1_MOSI_M1/SPI_SLV_MOSI/GPIO0_B2_u	GRF_GPIO0B_IOMUX_SEL_L[11:8] = 4'h2
vip_data7	I	LCD_D7/CIF_D7/UART2_TX_M1/SPI1_MISO_M1/SPI_SLV_MISO/GPIO0_B3_u	GRF_GPIO0B_IOMUX_SEL_L[15:12] = 4'h2

6.6 Application Notes

The biggest configuration requirement of all operations is the CAP_EN bit must be set after all the mode selection is ready. The configuration order of the input/output data format, YUV order, the address, frame size/width, AXI burst length and other options do not need to care.

There are many debug registers to make it easy to read the internal operation information of VICAP. The valid pixel number of scale result in FIFO can be known by read VICAP_DVP_FIFO_ ENTRY. The line number of last frame and the pixel number of last line can be also known by read the VICAP_LAST_LINE and VICAP_LAST_PIX.

In the Block Ping-Pong mode, Block0 end interrupt and Block1 end interrupt will appear alternately, and the status of Block0/1 will be setted alternately, and the block_id could be read if it is needed. If the user do not clear the status of Block0/1 in time, the interrupt of Block Error will generate, and the rest of the frame will not be received. The user could change the Block0/1 start address when Block0/1 end interrupt generate.

Chapter 7 Mobile Storage Host Controller

7.1 Overview

The Mobile Storage Host Controller is designed to support Secure Digital memory (SD-max version 3.01) with 1 bit, Multimedia Card(MMC-max version 4.51) with 1 bit.
The Host Controller supports following features:

- Bus Interface Features:
  - Support AMBA AHB interface for master and slave
  - Supports internal DMA interface(IDMAC)
    - Supports 16/32-bit data transfers
    - Single-channel; single engine used for Transmit and Receive, which are mutually exclusive
    - Dual-buffer and chained descriptor linked list
    - Each descriptor can transfer up to 4KB of data in chained mode and 8KB of data in dual-buffer mode
    - Programmable burst size for optimal host bus utilization
  - Support combined single FIFO for both transmit and receive operations
  - Support FIFO size of 256x32
  - Support FIFO over-run and under-run prevention by stopping card clock

- Card Interface Features:
  - Support Secure Digital memory protocol commands
  - Support Secure Digital I/O protocol commands
  - Support Multimedia Card protocol commands
  - Support Command Completion Signal and interrupts to host
  - Support CRC generation and error detection
  - Support programmable baud rate
  - Support power management and power switch
  - Support hardware reset
  - Support block size of 1 to 65,535 bytes
  - Support 1-bit SDR modes
  - Support boot in 1-bit SDR modes
  - Support Packed Commands, CMD21, CMD49

- Clock Interface Features:
  - Support 0/90/180/270-degree phase shift operation for sample clock (cclk_in_sample) and drive clock(cclk_in_drv) relative to function clock(cclk_in) respectively
  - Support phase tuning using delay line for sample clock(cclk_in_sample) and drive clock(cclk_in_drv) relative to function clock (cclk_in) respectively. The max number of delay element number is 256

7.2 Block Diagram

The Host Controller consists of the following main functional blocks.

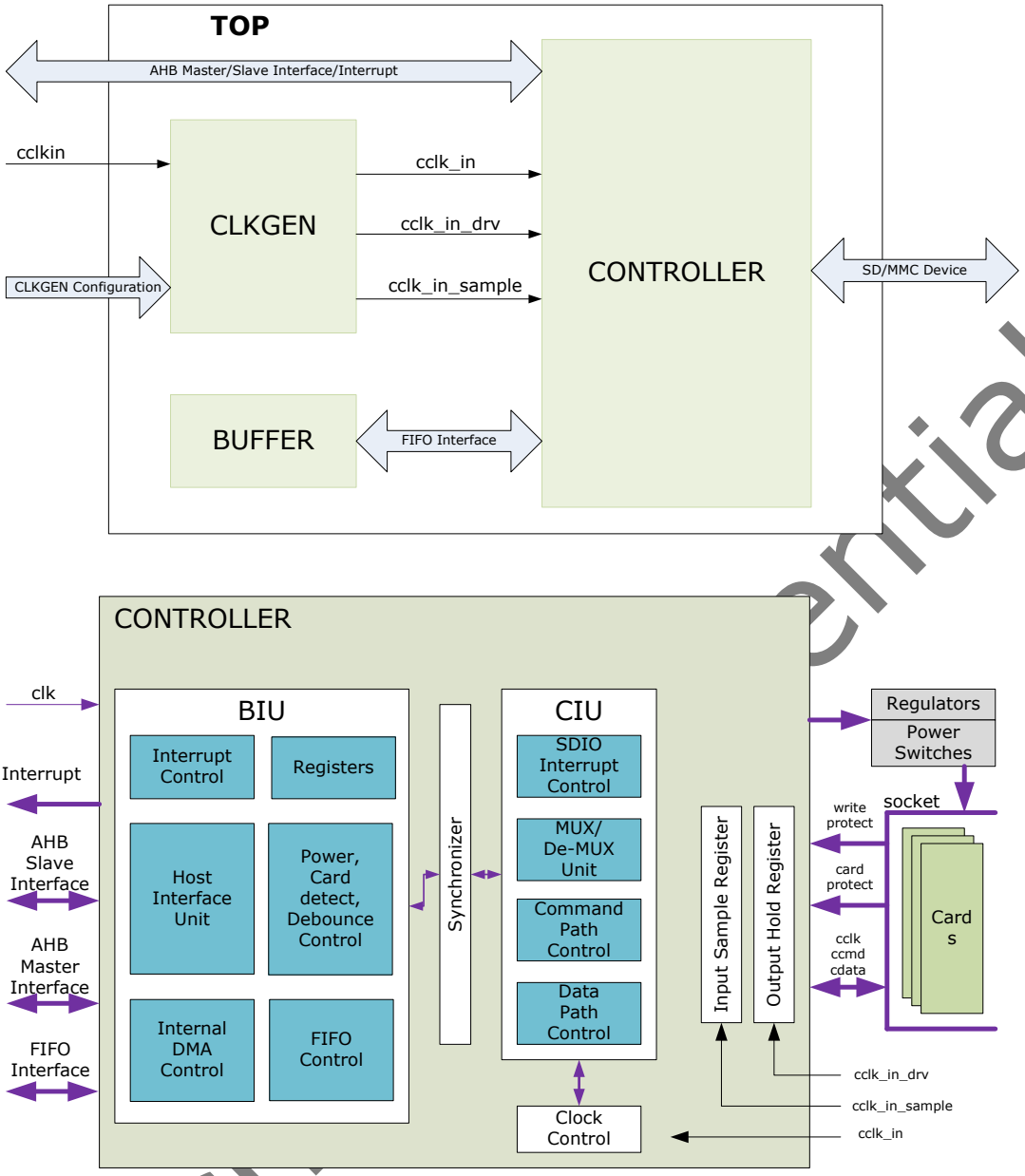

Fig. 7-1 Host Controller Block Diagram

- Clock Generate Unit(CLKGEN): generates card interface clock cclk_in/ cclk_sample/cclk_drv based on cclkin and configuration information.
- Asynchronous dual-port memory(BUFFER): Uses a two-clock synchronous read and synchronous write dual-port RAM. One of the ports is connected to the host clock, and the second port is connected to the card clock.
- Bus Interface Unit (BIU): Provides AMBA AHB interfaces for register and data read/writes.
- Card Interface Unit (CIU): Takes care of the SD/MMC protocols and provides clock management.

7.3 Function Description

7.3.1 Bus Interface Unit

The Bus Interface Unit provides the following functions:

- Host interface
- Interrupt control
- Register access

- External FIFO access
- Power control and card detection

1. Host Interface Unit

The Host Interface Unit is an AHB slave interface, which provides the interface between the SD/MMC card and the host bus.

2. Register Unit

The register unit is part of the bus interface unit; it provides read and write access to the registers.

All registers reside in the Bus Interface Unit clock domain. When a command is sent to a card by setting the start_bit, which is bit[31] of the SDMMC_CMD register, all relevant registers needed for the CIU operation are transferred to the CIU block. During this time, the registers that are transferred from the BIU to the CIU should not be written. The software should wait for the hardware to clear the start bit before writing to these registers again. The register unit has a hardware locking feature to prevent illegal writes to registers. The lock is necessary in order to avoid metastability violations, both because the host and card clock domains are different and to prevent illegal software operations.

Once a command start is issued by setting the start_bit of the SDMMC_CMD register, the following registers cannot be reprogrammed until the command is accepted by the card interface unit:

- SDMMC_CMD – Command
- SDMMC_CMDARG – Command Argument
- SDMMC_BYTCNT – Byte Count
- SDMMC_BLKSIZ – Block Size
- SDMMC_CLKDIV – Clock Divider
- SDMMC_CLKENA – Clock Enable
- SDMMC_CLKSRC – Clock Source
- SDMMC_TMOUT – Timeout
- SDMMC_CTYPE – Card Type

The hardware resets the start_bit once the CIU accepts the command. If a host write to any of these registers is attempted during this locked time, then the write is ignored and the hardware lock error bit is set in the raw interrupt status register. Additionally, if the interrupt is enabled and not masked for a hardware lock error, then an interrupt is sent to the host.

When the Card Interface Unit is in an idle state, it typically takes the following number of clocks for the command handshake, where clk is the BIU clock and cclk_in is the CIU clock: 3 (clk) + 3 (cclk_in)

Once a command is accepted, you can send another command to the CIU-which has a one-deep command queue-under the following conditions:

- If the previous command was not a data transfer command, the new command is sent to the SD/MMC card once the previous command completes.
- If the previous command is a data transfer command and if wait_prvdata_complete (bit[13]) of the Command register is set for the new command, the new command is sent to the SD/MMC card only when the data transfer completes.
- If the wait_prvdata_complete is 0, then the new command is sent to the SD/MMC card as soon as the previous command is sent. Typically, you should use this only to stop or abort a previous data transfer or query the card status in the middle of a data transfer.

3. Interrupt Controller Unit

The interrupt controller unit generates an interrupt that depends on the controller raw interrupt status, the interrupt-mask register, and the global interrupt-enable register bit. Once an interrupt condition is detected, it sets the corresponding interrupt bit in the raw

interrupt status register. The raw interrupt status bit stays on until the software clears the bit by writing a 1 to the interrupt bit; a 0 leaves the bit untouched.
The interrupt port, int, is an active-high, level-sensitive interrupt. The interrupt port is active only when any bit in the raw interrupt status register is active, the corresponding interrupt mask bit is 1, and the global interrupt enable bit is 1. The interrupt port is registered in order to avoid any combinational glitches.
The int_enable is reset to 0 on power-on, and the interrupt mask bits are set to 32'h0, which masks all the interrupts.
Notes:
Before enabling the interrupt, it is always recommended that you write 32'hffff_ffff to the raw interrupt status register in order to clear any pending unserviced interrupts. When clearing interrupts during normal operation, ensure that you clear only the interrupt bits that you serviced.
The SDIO Interrupts, Receive FIFO Data Request (RXDR), and Transmit FIFO Data Request (TXDR) are set by level-sensitive interrupt sources. Therefore, the interrupt source should be first cleared before you can clear the interrupt bit of the Raw Interrupt register. For example, on seeing the Receive FIFO Data Request (RXDR) interrupt, the FIFO should be emptied so that the "FIFO count greater than the RX-Watermark" condition, which triggers the interrupt, becomes inactive. The rest of the interrupts are triggered by a single clock-pulse-width source.

Table 7-1 Bits in Interrupt Status Register

Bits	Interrupt	Description
24	SDIO Interrupt	Interrupt from SDIO card. In MMC-Ver3.3-only mode, these bits are always 0
16	Card no-busy	If card exit busy status, the interrupt happened
15	End Bit Error (read) /Write no CRC (EBE)	Error in end-bit during read operation, or no data CRC received during write operation. For MMC CMD19, there may be no CRC status returned by the card. Hence, EBE is set for CMD19. The application should not treat this as an error.
14	Auto Command Done (ACD)	Stop/abort commands automatically sent by card unit and not initiated by host; similar to Command Done (CD) interrupt. Recommendation: Software typically need not enable this for non CE-ATA accesses; Data Transfer Over (DTO) interrupt that comes after this interrupt determines whether data transfer has correctly competed.
13	Start Bit Error (SBE)	Error in data start bit when data is read from a card. In 4-bit mode, if DAT[0] line indicates start bit-that is, 0-and any of the other data bits do not have start bit, then this error is set. Busy Complete Interrupt when data is written to the card. This interrupt is generated after completion of busy driven by the card after the last data block is written into the card.
12	Hardware Locked write Error (HLE)	During hardware-lock period, write attempted to one of locked registers. When software sets the start_cmd bit in the SDMMC_CMD register, the Host Controller tries to load the command. If the command buffer is already filled with a command, this error is raised. The software then has to reload the command.
11	FIFO Underrun/ Overrun Error (FRUN)	Host tried to push data when FIFO was full, or host tried to read data when FIFO was empty. Typically this should not happen, except due to error in software. Card unit never pushes data into FIFO when FIFO is full, and pop data when FIFO is empty. If IDMAC is enabled, FIFO underrun/overrun can occur due to a programming error on MSIZE and

Bits	Interrupt	Description
		watermark values in SDMMC_FIFOTH register.
10	Data Starvation by Host Timeout (HTO)	To avoid data loss, card clock out is stopped if FIFO is empty when writing to card, or FIFO is full when reading from card. Whenever card clock is stopped to avoid data loss, data-starvation timeout counter is started with data-timeout value. This interrupt is set if host does not fill data into FIFO during write to card, or does not read from FIFO during read from card before timeout period. Even after timeout, card clock stays in stopped state, with CIU state machines waiting. It is responsibility of host to push or pop data into FIFO upon interrupt, which automatically restarts cclk_out and card state machines. Even if host wants to send stop/abort command, it still needs to ensure it has to push or pop FIFO so that clock starts in order for stop/abort command to send on cmd signal along with data that is sent or received on data line.
9	Data Read Timeout (DRTO)	In Normal functioning mode: Data read timeout (DRTO) Data timeout occurred. Data Transfer Over (DTO) also set if data timeout occurs. In Boot Mode: Boot Data Start (BDS) When set, indicates that Host Controller has started to receive boot data from the card. A write to this register with a value of 1 clears this interrupt.
8	Response Timeout (RTO)	In normal functioning mode: Response timeout (RTO) Response timeout occurred. Command Done (CD) also set if response timeout occurs. If command involves data transfer and when response times out, no data transfer is attempted by Host Controller. In Boot Mode: Boot Ack Received (BAR) When expect_boot_ack is set, on reception of a boot acknowledge pattern—0-1-0—this interrupt is asserted. A write to this register with a value of 1 clears this interrupt.
7	Data CRC Error (DCRC)	Received Data CRC does not match with locally-generated CRC in CIU. Can also occur if the Write CRC status is incorrectly sampled by the Host.
6	Response CRC Error (RCRC)	Response CRC does not match with locally-generated CRC in CIU.
5	Receive FIFO Data Request (RXDR)	Interrupt set during read operation from card when FIFO level is greater than Receive-Threshold level. Recommendation: In DMA modes, this interrupt should not be enabled. In non-DMA mode: pop RX_WMark + 1 data from FIFO.
4	Transmit FIFO Data Request (TXDR)	Interrupt set during write operation to card when FIFO level reaches less than or equal to Transmit-Threshold level. Recommendation: In DMA modes, this interrupt should not be enabled. In non-DMA mode: if (pending_bytes > (FIFO_DEPTH - TX_WMark))

Bits	Interrupt	Description
		push (FIFO_DEPTH - TX_WMark) data into FIFO else push pending_bytes data into FIFO
3	Data Transfer Over (DTO)	Indicates Data transfer completed. Though on detection of errors-Start Bit Error, Data CRC error, and so on, DTO may or may not be set; the application must issue CMD12, which ensures that DTO is set. Recommendation: In non-DMA mode, when data is read from card, on seeing interrupt, host should read any pending data from FIFO. In DMA mode, DMA controllers guarantee FIFO is flushed before interrupt. DTO bit is set at the end of the last data block, even if the device asserts MMC busy after the last data block.
2	Command Done(CD)	Command sent to card and got response from card, even if Response Error or CRC error occurs.
1	Response Error (RE)	Error in received response set if one of following occurs: ● Transmission bit != 0 ● Command index mismatch ● End-bit != 1
0	Card-Detect (CDT)	When one or more cards inserted or removed, this interrupt occurs. Software should read card-detect register to determine current card status. Recommendation: After power-on and before enabling interrupts, software should read card detect register and store it in memory. When interrupt occurs, it should read card detect register and compare it with value stored in memory to determine which card(s) were removed/inserted. Before exiting ISR, software should update memory with new card-detect value.

4. FIFO Controller Unit

The FIFO controller interfaces the external FIFO to the host interface and the card controller unit. When FIFO overrun and under-run conditions occur, the card clock stops in order to avoid data loss.

The FIFO uses a two-clock synchronous read and synchronous write dual-port RAM. One of the ports is connected to the host clock, clk, and the second port is connected to the card clock, cclk_in.

Notes: The FIFO controller does not support simultaneous read/write access from the same port. For debugging purposes, the software may try to write into the FIFO and read back the data; results are indeterminate, since the design does not support read/write access from the same port.

5. Power Control and Card Detection Unit

The register unit has registers that control the power. Power to each card can be selectively turned on or off.

The card detection unit looks for any changes in the card-detect signals for card insertion or card removal. It filters out the debounces associated with mechanical insertion or removal, and generates one interrupt to the host. You can program the debounce filter value.

On power-on, the controller should read in the card_detect port and store the value in the memory. Upon receiving a card-detect interrupt, it should again read the card_detect port and XOR with the previous card-detect status to find out which card has interrupted. If

more than one card is simultaneously removed or inserted, there is only one card-detect interrupt; the XOR value indicates which cards have been disturbed. The memory should be updated with the new card-detect value.

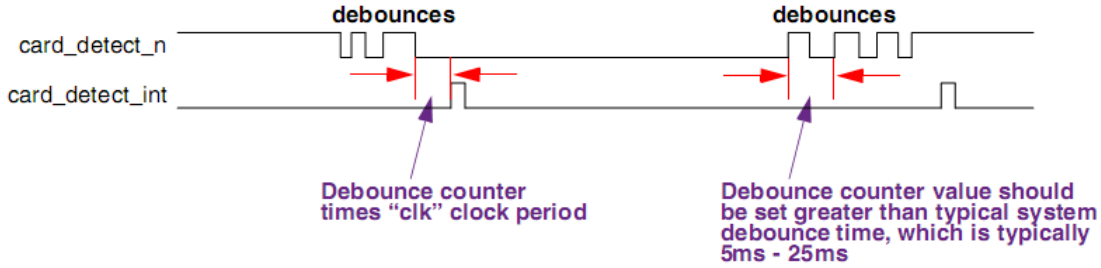

Fig. 7-2 SD/MMC Card-Detect Signal

6. DMA Interface Unit

DMA signals interface the Host Controller to an external DMA controller to reduce the software overhead during FIFO data transfers. The DMA request/acknowledge handshake is used for only data transfers. The DMA interface provides a connection to the DMA Controller.

On seeing the DMA request, the DMA controller initiates accesses through the host interface to read or write into the data FIFO. The Host Controller has FIFO transmit/receive watermark registers that you can set, depending on system latency. The DMA interface asserts the request in the following cases:

- Read from a card when the data FIFO word count exceeds the Rx-Watermark level
- Write to a card when the FIFO word count is less than or equal to the Tx-Watermark level

When the DMA interface is enabled, you can use normal host read/write to access the data FIFO.

7.3.2 Card Interface Unit

The Card Interface Unit (CIU) interfaces with the Bus Interface Unit (BIU) and the devices. The host writes command parameters to the BIU control registers, and these parameters are then passed to the CIU. Depending on control register values, the CIU generates SD/MMC command and data traffic on a selected card bus according to SD/MMC protocol. The Host Controller accordingly controls the command and data path.

The following software restrictions should be met for proper CIU operation:

- Only one data transfer command can be issued at a time.
- During an open-ended card write operation, if the card clock is stopped because the FIFO is empty, the software must first fill the data into the FIFO and start the card clock. It can then issue only a stop/abort command to the card.
- When issuing card reset commands (CMD0, CMD15 or CMD52_reset) while a card data transfer is in progress, the software must set the stop_abort_cmd bit in the Command register so that the Host Controller can stop the data transfer after issuing the card reset command.
- When the data end bit error is set in the SDMMC_RINTSTS register, the Host Controller does not guarantee SDIO interrupts. The software should ignore the SDIO interrupts and issue the stop/abort command to the card, so that the card stops sending the read data.
- If the card clock is stopped because the FIFO is full during a card read, the software should read at least two FIFO locations to start the card clock.

The CIU block consists of the following primary functional blocks:

- Command path
- Data path
- SDIO interrupt control

- Clock control
- Mux/demux unit

1. Command Path

The command path performs the following functions:

- Loads clock parameters
- Loads card command parameters
- Sends commands to card bus (ccmd_out line)
- Receives responses from card bus (ccmd_in line)
- Sends responses to BIU
- Drives the P-bit on command line

A new command is issued to the Host Controller by programming the BIU registers and setting the start_cmd bit in the Command register. The BIU asserts start_cmd, which indicates that a new command is issued to the SD/MMC device. The command path loads this new command (command, command argument, timeout) and sends acknowledge to the BIU by asserting cmd_taken.

Once the new command is loaded, the command path state machine sends a command to the device bus-including the internally generated CRC7-and receives a response, if any. The state machine then sends the received response and signals to the BIU that the command is done, and then waits for eight clocks before loading a new command.

Load Command Parameters

One of the following commands or responses is loaded in the command path:

- New command from BIU – When start_cmd is asserted, then the start_cmd bit is set in the Command register.
- Internally-generated auto-stop command – When the data path ends, the stop command request is loaded.
- IRQ response with RCA 0x000 – When the command path is waiting for an IRQ response from the MMC card and a "send irq response" request is signaled by the BIU, then the send_irq_response bit is set in the control register.

Loading a new command from the BIU in the command path depends on the following Command register bit settings:

- update_clock_registers_only – If this bit is set in the Command register, the command path updates only the clock enable, clock divider, and clock source registers. If this bit is not set, the command path loads the command, command argument, and timeout registers; it then starts processing the new command.
- wait_prvdata_complete – If this bit is set, the command path loads the new command under one of the following conditions:
  - Immediately, if the data path is free (that is, there is no data transfer in progress), or if an open-ended data transfer is in progress (byte_count = 0).
  - After completion of the current data transfer, if a predefined data transfer is in progress.

Send Command and Receive Response

Once a new command is loaded in the command path, update_clock_registers_only bit is unset – the command path state machine sends out a command on the device bus; the command path state machine is illustrated in following figure.

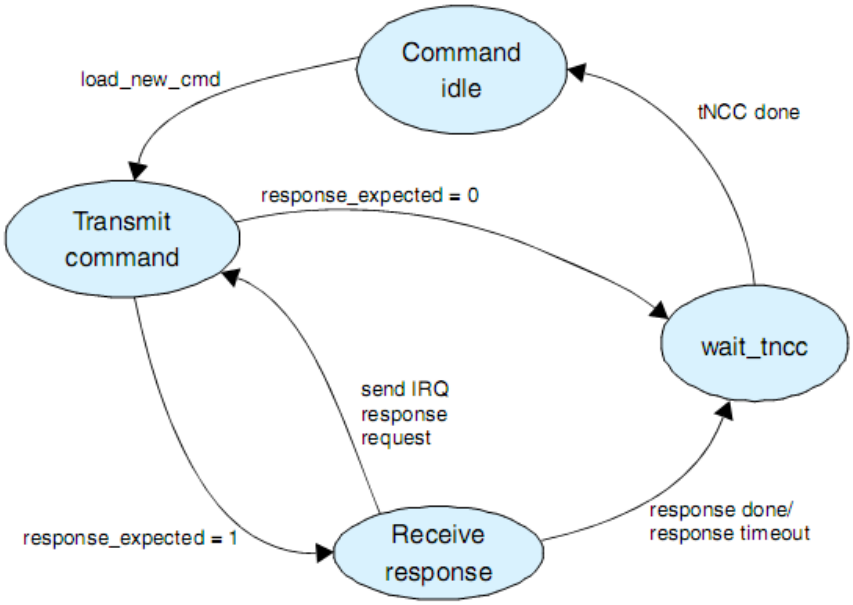

Fig. 7-3 Host Controller Command Path State Machine

The command path state machine performs the following functions, according to Command register bit values:

- send_initialization – Initialization sequence of 80 clocks is sent before sending the command.
- response_expected – Response is expected for the command. After the command is sent out, the command path state machine receives a 48-bit or 136-bit response and sends it to the BIU. If the start bit of the card response is not received within the number of clocks programmed in the timeout register, then the response timeout and command done bit is set in the Raw Interrupt Status register as a signal to the BIU. If the response-expected bit is not set, the command path sends out a command and signals a response done to the BIU; that is, the command done bit is set in the Raw Interrupt Status register.
- response_length – If this bit is set, a 136-bit response is received; if it is not set, a 48-bit response is received.
- check_response_crc – If this bit is set, the command path compares CRC7 received in the response with the internally-generated CRC7. If the two do not match, the response CRC error is signaled to the BIU; that is, the response CRC error bit is set in the Raw Interrupt Status register.

Send Response to BIU

If the response_expected bit is set in the Command register, the received response is sent to the BIU. The Response0 register is updated for a short response, and the Response3, Response2, Response1, and Response0 registers are updated on a long response, after which the Command Done bit is set. If the response is for an auto_stop command sent by the CIU, the response is saved in the Response1 register, after which the Auto Command Done bit is set.

Additionally, the command path checks for the following:

- Transmission bit = 0
- Command index matches command index of the sent command
- End bit = 1 in received card response

The command index is not checked for a 136-bit response or if the check_response_crc bit is unset. For a 136-bit response and reserved CRC 48-bit responses, the command index is reserved-that is, 111111.

Polling Command Completion Signal

The device generates the Command Completion Signal in order to notify the host controller

of the normal command completion or command termination.

Command Completion Signal Detection and Interrupt to Host Processor

If the ccs_expected bit is set in the Command register, the Command Completion Signal (CCS) from the device is indicated by setting the Data Transfer Over (DTO) bit in the SDMMC_RINTSTS register. The Host Controller generates a Data Transfer Over (DTO) interrupt if this interrupt is not masked.

Command Completion Signal Timeout

If the command expects a CCS from the device—if the ccs_expected bit is set in the Command register—the command state machine waits for the CCS and remains in a wait_CCSS state. If the device fails to send out the CCS, the host software should implement a timeout mechanism to free the command and data path. The host controller does not implement a hardware timer; it is the responsibility of the host software to maintain a software timer.

In the event of a CCS timeout, the host should issue a CCSD by setting the send_ccsd bit in the CTRL register. The host controller command state machine sends the CCSD to the device and exits to an idle state. After sending the CCSD, the host should also send a CMD12 to the device in order to abort the outstanding command.

Send Command Completion Signal Disable

If the send_ccsd bit is set in the CTRL register, the host sends a Command Completion Signal Disable (CCSD) pattern on the CMD line. The host can send the CCSD while waiting for the CCS or after a CCS timeout happens.

After sending the CCSD pattern, the host sets the Command Done (CD) bit in SDMMC_RINTSTS and also generates an interrupt to the host if the Command Done interrupt is not masked.

2. Data Path

The data path block pops the data FIFO and transmits data on cdata_out during a write data transfer, or it receives data on cdata_in and pushes it into the FIFO during a read data transfer. The data path loads new data parameters—that is, data expected, read/write data transfer, stream/block transfer, block size, byte count, card type, timeout registers—whenever a data transfer command is not in progress.

If the data_expected bit is set in the Command register, the new command is a data transfer command and the data path starts one of the following:

- Transmit data if the read/write bit = 1
- Data receive if read/write bit = 0

Data Transmit

The data transmit state machine, illustrated in following figure, starts data transmission two clocks after a response for the data write command is received; this occurs even if the command path detects a response error or response CRC error. If a response is not received from the card because of a response timeout, data is not transmitted. Depending upon the value of the transfer_mode bit in the Command register, the data transmit state machine puts data on the card data bus in a stream or in block(s).

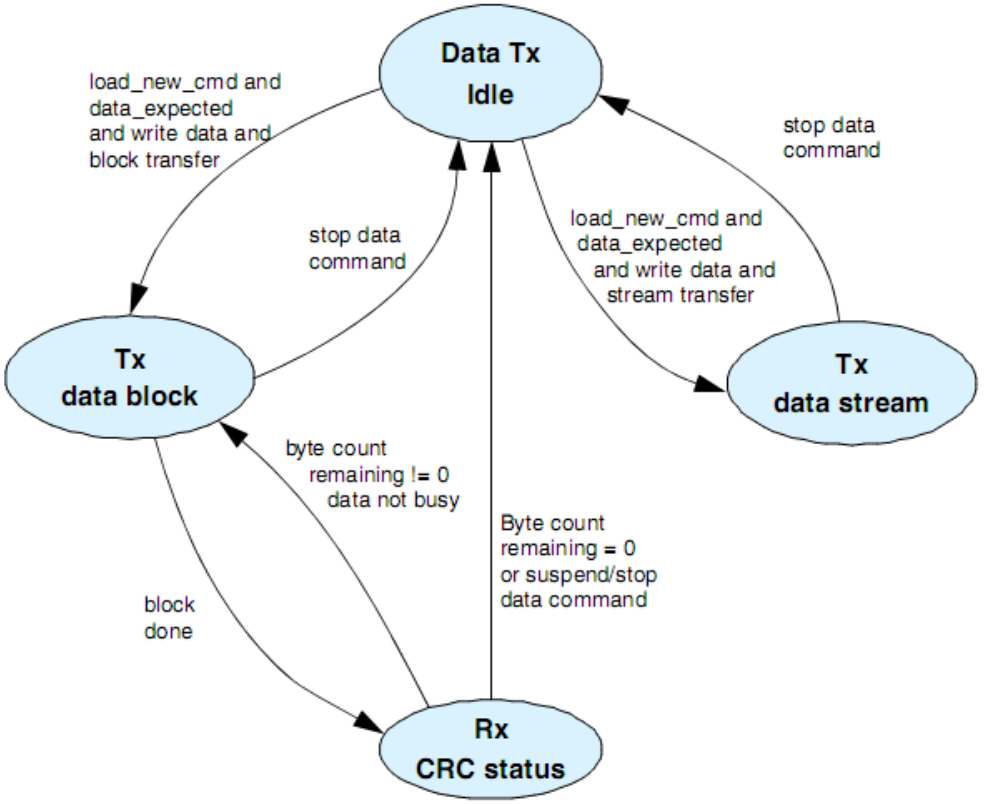

Fig. 7-4 Host Controller Data Transmit State Machine

Stream Data Transmit

If the transfer_mode bit in the Command register is set to 1, it is a stream-write data transfer. The data path pops the FIFO from the BIU and transmits in a stream to the card data bus. If the FIFO becomes empty, the card clock is stopped and restarted once data is available in the FIFO.

If the byte_count register is programmed to 0, it is an open-ended stream-write data transfer. During this data transfer, the data path continuously transmits data in a stream until the host software issues a stop command. A stream data transfer is terminated when the end bit of the stop command and end bit of the data match over two clocks.

If the byte_count register is programmed with a non-zero value and the send_auto_stop bit is set in the Command register, the stop command is internally generated and loaded in the command path when the end bit of the stop command occurs after the last byte of the stream write transfer matches.

This data transfer can also terminate if the host issues a stop command before all the data bytes are transferred to the card bus.

Single Block Data

If the transfer_mode bit in the Command register is set to 0 and the byte_count register value is equal to the value of the block_size register, a single-block write-data transfer occurs. The data transmit state machine sends data in a single block, where the number of bytes equals the block size, including the internally-generated CRC16.

If the SDMMC_CTYPE register bit for the selected card – indicated by the card_num value in the Command register – is set for a 1-bit, 4-bit, or 8-bit data transfer, the data is transmitted on 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and transmitted for 1, 4, or 8 data lines, respectively.

After a single data block is transmitted, the data transmit state machine receives the CRC status from the card and signals a data transfer to the BIU; this happens when the data-transfer-over bit is set in the SDMMC_RINTSTS register.

If a negative CRC status is received from the card, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the SDMMC_RINTSTS register.

Additionally, if the start bit of the CRC status is not received by two clocks after the end of the data block, a CRC status start bit error is signaled to the BIU by setting the write-no-CRC bit in the SDMMC_RINTSTS register.

Multiple Block Data

A multiple-block write-data transfer occurs if the transfer_mode bit in the Command register is set to 0 and the value in the byte_count register is not equal to the value of the block_size register. The data transmit state machine sends data in blocks, where the number of bytes in a block equals the block size, including the internally-generated CRC16.

If the SDMMC_CTYPE register bit for the selected card – indicated by the card_num value in the Command register – is set to 1-bit, 4-bit, or 8-bit data transfer, the data is transmitted on 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and transmitted on 1, 4, or 8 data lines, respectively.

After one data block is transmitted, the data transmit state machine receives the CRC status from the card. If the remaining byte_count becomes 0, the data path signals to the BIU that the data transfer is done; this happens when the data-transfer-over bit is set in the SDMMC_RINTSTS register.

If the remaining data bytes are greater than 0, the data path state machine starts to transmit another data block.

If a negative CRC status is received from the card, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the SDMMC_RINTSTS register, and continues further data transmission until all the bytes are transmitted.

Additionally, if the CRC status start bit is not received by two clocks after the end of a data block, a CRC status start bit error is signaled to the BIU by setting the write-no-CRC bit in the SDMMC_RINTSTS register; further data transfer is terminated.

If the send_auto_stop bit is set in the Command register, the stop command is internally generated during the transfer of the last data block, where no extra bytes are transferred to the card. The end bit of the stop command may not exactly match the end bit of the CRC status in the last data block.

If the block size is less than 4, 16, or 32 for card data widths of 1 bit, 4 bits, or 8 bits, respectively, the data transmit state machine terminates the data transfer when all the data is transferred, at which time the internally generated stop command is loaded in the command path.

If the byte_count is 0 – the block size must be greater than 0 – it is an open-ended block transfer. The data transmit state machine for this type of data transfer continues the block-write data transfer until the host software issues a stop or abort command.

Data Receive

The data-receive state machine, illustrated in following figure, receives data two clock cycles after the end bit of a data read command, even if the command path detects a response error or response CRC error. If a response is not received from the card because a response timeout occurs, the BIU does not receive a signal that the data transfer is complete; this happens if the command sent by the Host Controller is an illegal operation for the card, which keeps the card from starting a read data transfer.

If data is not received before the data timeout, the data path signals a data timeout to the BIU and an end to the data transfer done. Based on the value of the transfer_mode bit in the Command register, the data-receive state machine gets data from the card data bus in a stream or block(s).

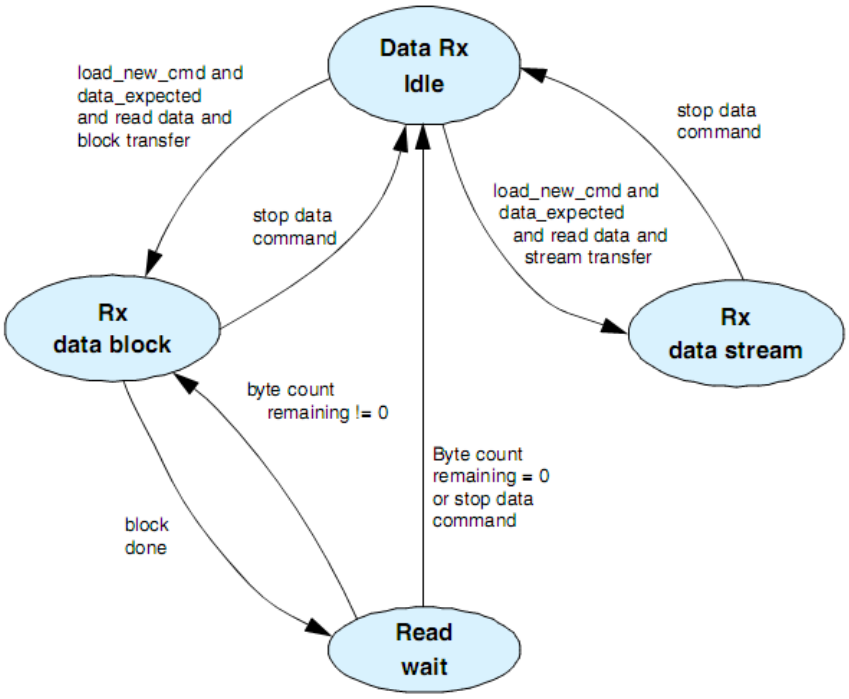

Fig. 7-5 Host Controller Data Receive State Machine

Stream Data Read

A stream-read data transfer occurs if the transfer_mode bit in the Command register equals 1, at which time the data path receives data from the card and pushes it to the FIFO. If the FIFO becomes full, the card clock stops and restarts once the FIFO is no longer full.

An open-ended stream-read data transfer occurs if the byte_count register equals 0. During this type of data transfer, the data path continuously receives data in a stream until the host software issues a stop command. A stream data transfer terminates two clock cycles after the end bit of the stop command.

If the byte_count register contains a non-zero value and the send_auto_stop bit is set in the Command register, a stop command is internally generated and loaded into the command path, where the end bit of the stop command occurs after the last byte of the stream data transfer is received. This data transfer can terminate if the host issues a stop or abort command before all the data bytes are received from the card.

Single-Block Data Read

A single-block read-data transfer occurs if the transfer_mode bit in the Command register is set to 0 and the value of the byte_count register is equal to the value of the block_size register. When a start bit is received before the data times out, data bytes equal to the block size and CRC16 are received and checked with the internally-generated CRC16.

If the SDMMC_CTYPE register bit for the selected card – indicated by the card_num value in the Command register – is set to a 1-bit, 4-bit, or 8-bit data transfer, data is received from 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and checked for 1, 4, or 8 data lines, respectively. If there is a CRC16 mismatch, the data path signals a data CRC error to the BIU. If the received end bit is not 1, the BIU receives an end-bit error.

Multiple-Block Data Read

If the transfer_mode bit in the Command register is set to 0 and the value of the byte_count register is not equal to the value of the block_size register, it is a multiple-block read-data transfer. The data-receive state machine receives data in blocks, where the number of bytes in a block is equal to the block size, including the internally-generated CRC16.

If the SDMMC_CTYPE register bit for the selected card – indicated by the card_num value in the Command register – is set to a 1-bit, 4-bit, or 8-bit data transfer, data is received from 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and checked for 1, 4, or 8 data lines, respectively.

After a data block is received, if the remaining byte_count becomes 0, the data path signals a data transfer to the BIU.

If the remaining data bytes are greater than 0, the data path state machine causes another data block to be received. If CRC16 of a received data block does not match the internally-generated CRC16, a data CRC error to the BIU and data reception continue further data transmission until all bytes are transmitted.
Additionally, if the end of a received data block is not 1, data on the data path signals terminate the bit error to the CIU and the data-receive state machine terminates data reception, waits for data timeout, and signals to the BIU that the data transfer is complete.
If the send_auto_stop bit is set in the Command register, the stop command is internally generated when the last data block is transferred, where no extra bytes are transferred from the card; the end bit of the stop command may not exactly match the end bit of the last data block.
If the requested block size for data transfers to cards is less than 4, 16, or 32 bytes for 1-bit, 4-bit, or 8-bit data transfer modes, respectively, the data-transmit state machine terminates the data transfer when all data is transferred, at which point the internally-generated stop command is loaded in the command path. Data received from the card after that are then ignored by the data path.
If the byte_count is 0—the block size must be greater than 0—it is an open-ended block transfer. For this type of data transfer, the data-receive state machine continues the block-read data transfer until the host software issues a stop or abort command.

Auto-Stop

The Host Controller internally generates a stop command and is loaded in the command path when the send_auto_stop bit is set in the Command register.
The software should set the send_auto_stop bit according to details listed in following table.

Table 7-2 Auto-Stop Generation

Card type	Transfer type	Byte Count	send_auto_stop bit set	Comments
MMC	Stream read	0	No	Open-ended stream
MMC	Stream read	>0	Yes	Auto-stop after all bytes transfer
MMC	Stream write	0	No	Open-ended stream
MMC	Stream write	>0	Yes	Auto-stop after all bytes transfer
MMC	Single-block read	>0	No	Byte count =0 is illegal
MMC	Single-block write	>0	No	Byte count =0 is illegal
MMC	Multiple-block read	0	No	Open-ended multiple block
MMC	Multiple-block read	>0	Yes①	Pre-defined multiple block
MMC	Multiple-block write	0	No	Open-ended multiple block
MMC	Multiple-block write	>0	Yes①	Pre-defined multiple block
SDMEM	Single-block read	>0	No	Byte count =0 is illegal
SDMEM	Single-block write	>0	No	Byte count =0 illegal
SDMEM	Multiple-block read	0	No	Open-ended multiple block
SDMEM	Multiple-block read	>0	Yes	Auto-stop after all bytes transfer
SDMEM	Multiple-block write	0	No	Open-ended multiple block
SDMEM	Multiple-block write	>0	Yes	Auto-stop after all bytes transfer
SDIO	Single-block read	>0	No	Byte count =0 is illegal
SDIO	Single-block write	>0	No	Byte count =0 illegal
SDIO	Multiple-block read	0	No	Open-ended multiple block

Card type	Transfer type	Byte Count	send_auto_stop bit set	Comments
SDIO	Multiple-block read	>0	No	Pre-defined multiple block
SDIO	Multiple-block write	0	No	Open-ended multiple block
SDIO	Multiple-block write	>0	No	Pre-defined multiple block

①:*The condition under which the transfer mode is set to block transfer and byte_count is equal to block size is treated as a single-block data transfer command for both MMC and SD cards. If byte_count = n*block_size (n = 2, 3, …), the condition is treated as a predefined multiple-block data transfer command. In the case of an MMC card, the host software can perform a predefined data transfer in two ways: 1) Issue the CMD23 command before issuing CMD18/CMD25 commands to the card – in this case, issue MD18/CMD25 commands without setting the send_auto_stop bit. 2) Issue CMD18/CMD25 commands without issuing CMD23 command to the card, with the send_auto_stop bit set. In this case, the multiple-block data transfer is terminated by an internally-generated auto-stop command after the programmed byte count.*

The following list conditions for the auto-stop command.

- Stream read for MMC card with byte count greater than 0 – The Host Controller generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent out when the last byte of data is read from the card and no extra data byte is received. If the byte count is less than 6 (48 bits), a few extra data bytes are received from the card before the end bit of the stop command is sent.
- Stream write for MMC card with byte count greater than 0 - The Host Controller generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent when the last byte of data is transmitted on the card bus and no extra data byte is transmitted. If the byte count is less than 6 (48 bits), the data path transmits the data last in order to meet the above condition.
- Multiple-block read memory for SD card with byte count greater than 0 – If the block size is less than 4 (single-bit data bus), 16 (4-bit data bus), or 32 (8-bit data bus), the auto-stop command is loaded in the command path after all the bytes are read. Otherwise, the top command is loaded in the command path so that the end bit of the stop command is sent after the last data block is received.
- Multiple-block write memory for SD card with byte count greater than 0 – If the block size is less than 3 (single-bit data bus), 12 (4-bit data bus), or 24 (8-bit data bus), the auto-stop command is loaded in the command path after all data blocks are transmitted. Otherwise, the stop command is loaded in the command path so that the end bit of the stop command is sent after the end bit of the CRC status is received.
- Precaution for host software during auto-stop – Whenever an auto-stop command is issued, the host software should not issue a new command to the SD/MMC device until the auto-stop is sent by the Host Controller and the data transfer is complete. If the host issues a new command during a data transfer with the auto-stop in progress, an auto-stop command may be sent after the new command is sent and its response is received; this can delay sending the stop command, which transfers extra data bytes. For a stream write, extra data bytes are erroneous data that can corrupt the card data. If the host wants to terminate the data transfer before the data transfer is complete, it can issue a stop or abort command, in which case the Host Controller does not generate an auto-stop command.

3. Non-Data Transfer Commands that Use Data Path

Some non-data transfer commands (non-read/write commands) also use the data path.

Following table lists the commands and register programming requirements for them.

Table 7-3 Non-data Transfer Commands and Requirements

Base Address [12:8]	CMD 27	CMD 30	CMD 42	ACMD 13	ACMD 22	ACMD 51
Command register programming						
cmd_index	6'h1B	6'h1E	6'h2A	6'h0D	6'h16	6'h33
response_expect	1	1	1	1	1	1
rResponse_length	0	0	0	0	0	0
check_response_crc	1	1	1	1	1	1
data_expected	1	1	1	1	1	1
read/write	1	0	1	0	0	0
transfer_mode	0	0	0	0	0	0
send_auto_stop	0	0	0	0	0	0
wait_prevdata_complete	0	0	0	0	0	0
stop_abort_cmd	0	0	0	0	0	0
Command Argument register programming						
	stuff bits	32-bit write protect data address	stuff bits	stuff bits	stuff bits	stuff bits
Block Size register programming						
	16	4	Num_bytes①	64	4	8
Byte Count register programming						
	16	4	Num_bytes①	64	4	8

①: *Num_bytes = No. of bytes specified as per the lock card data structure (Refer to the SD specification and the MMC specification)*

4. SDIO Interrupt Control

Interrupts for SD cards are reported to the BIU by asserting an interrupt signal for two clock cycles. SDIO cards signal an interrupt by asserting cdata_in low during the interrupt period; an interrupt period for the selected card is determined by the interrupt control state machine. An interrupt period is always valid for non-active or non-selected cards, and 1-bit data mode for the selected card. An interrupt period for a wide-bus active or selected card is valid for the following conditions:

- Card is idle
- Non-data transfer command in progress
- Third clock after end bit of data block between two data blocks
- From two clocks after end bit of last data until end bit of next data transfer command

Bear in mind that, in the following situations, the controller does not sample the SDIO interrupt of the selected card when the card data width is 4 bits. Since the SDIO interrupt is level-triggered, it is sampled in a further interrupt period and the host does not lose any SDIO interrupt from the card.

- Read/Write Resume – The CIU treats the resume command as a normal data transfer command. SDIO interrupts during the resume command are handled similarly to other data commands. According to the SDIO specification, for the normal data command the interrupt period ends after the command end bit of the data command; for the resume command, it ends after the response end bit. In the case of the resume command, the Controller stops the interrupt sampling period after the resume command end bit, instead of stopping after the response end bit of the resume command.
- Suspend during read transfer – If the read data transfer is suspended by the host, the host sets the abort_read_data bit in the controller to reset the data state machine. In the CIU, the SDIO interrupts are handled such that the interrupt sampling starts after the abort_read_data bit is set by the host. In this case the controller does not sample

SDIO interrupts between the period from response of the suspend command to setting the abort_read_data bit, and starts sampling after setting the abort_read_data bit.

5. Clock Control

The clock control block provides different clock frequencies required for SD/MMC cards. The cclk_in signal is the source clock (cclk_in >= card max operating frequency) for clock divider of the clock control block. This source clock (cclk_in) is used to generate different card clock frequencies (cclk_out). The card clock can have different clock frequencies, since the card can be a low-speed card or a full-speed card. The Host Controller provides one clock signal (cclk_out).

The clock frequency of a card depends on the following clock control registers:

- Clock Divider register – Internal clock dividers are used to generate different clock frequencies required for card. The division factor for each clock divider can be programmed by writing to the Clock Divider register. The clock divider is an 8-bit value that provides a clock division factor from 1 to 510; a value of 0 represents a clock-divider bypass, a value of 1 represents a divide by 2, a value of 2 represents a divide by 4, and so on.
- Clock Control register – cclk_out can be enabled or disabled for each card under the following conditions:
  - clk_enable – cclk_out for a card is enabled if the clk_enable bit for a card in the Clock Control register is programmed (set to 1) or disabled (set to 0).
  - Low-power mode – Low-power mode of a card can be enabled by setting the low-power mode bit of the Clock Control register to 1. If low-power mode is enabled to save card power, the cclk_out is disabled when the card is idle for at least 8 card clock cycles. It is enabled when a new command is loaded and the command path goes to a non-idle state.

Additionally, cclk_out is disabled when an internal FIFO is full – card read (no more data can be received from card) – or when the FIFO is empty – card write (no data is available for transmission). This helps to avoid FIFO overrun and underrun conditions. It is used by the command and data path to qualify cclk_in for driving outputs and sampling inputs at the programmed clock frequency for the selected card, according to the Clock Divider and Clock Source register values.

Under the following conditions, the card clock is stopped or disabled, along with the active clk_en, for the selected card:

- Clock can be disabled by writing to Clock Enable register (clk_en bit = 1).
- If low-power mode is selected and card is idle, or not selected for 8 clocks.
- FIFO is full and data path cannot accept more data from the card and data transfer is incomplete –to avoid FIFO overrun.
- FIFO is empty and data path cannot transmit more data to the card and data transfer is incomplete – to avoid FIFO underrun.

6. Error Detection

- Response
  - Response timeout – Response expected with response start bit is not received within programmed number of clocks in timeout register.
  - Response CRC error – Response is expected and check response CRC requested; response CRC7 does not match with the internally-generated CRC7.
  - Response error – Response transmission bit is not 0, command index does not match with the command index of the send command, or response end bit is not 1.
- Data transmit

- No CRC status – During a write data transfer, if the CRC status start bit is not received two clocks after the end bit of the data block is sent out, the data path does the following:
  - Signals no CRC status error to the BIU
  - Terminates further data transfer
  - Signals data transfer done to the BIU
- Negative CRC – If the CRC status received after the write data block is negative (that is, not 010), a data CRC error is signaled to the BIU and further data transfer is continued.
- Data starvation due to empty FIFO – If the FIFO becomes empty during a write data transmission, or if the card clock is stopped and the FIFO remains empty for data timeout clocks, then a data-starvation error is signaled to the BIU and the data path continues to wait for data in the FIFO.

- Data receive
  - Data timeout – During a read-data transfer, if the data start bit is not received before the number of clocks that were programmed in the timeout register, the data path does the following:
    - Signals data-timeout error to the BIU
    - Terminates further data transfer
    - Signals data transfer done to BIU
  - Data start bit error – During a 4-bit or 8-bit read-data transfer, if the all-bit data line does not have a start bit, the data path signals a data start bit error to the BIU and waits for a data timeout, after which it signals that the data transfer is done.
  - Data CRC error – During a read-data-block transfer, if the CRC16 received does not match with the internally generated CRC16, the data path signals a data CRC error to the BIU and continues further data transfer.
  - Data end-bit error – During a read-data transfer, if the end bit of the received data is not 1, the data path signals an end-bit error to the BIU, terminates further data transfer, and signals to the BIU that the data transfer is done.
  - Data starvation due to FIFO full – During a read data transmission and when the FIFO becomes full, the card clock is stopped. If the FIFO remains full for data timeout clocks, a data starvation error is signaled to the BIU (Data Starvation by Host Timeout bit is set in SDMMC_RINTSTS register) and the data path continues to wait for the FIFO to start to empty.

7.3.3 Internal Direct Memory Access Controller (IDMAC)

The Internal Direct Memory Access Controller (IDMAC) has a Control and Status Register (CSR) and a single Transmit/Receive engine, which transfers data from host memory to the device port and vice versa. The controller utilizes a descriptor to efficiently move data from source to destination with minimal Host CPU intervention. You can program the controller to interrupt the Host CPU in situations such as data Transmit and Receive transfer completion from the card, as well as other normal or error conditions.
The IDMAC and the Host driver communicate through a single data structure. CSR addresses 0x80 to 0x98 are reserved for host programming.
The IDMAC transfers the data received from the card to the Data Buffer in the Host memory, and it transfers Transmit data from the Data Buffer in the Host memory to the FIFO. Descriptors that reside in the Host memory act as pointers to these buffers.
A data buffer resides in physical memory space of the Host and consists of complete data or partial data. Buffers contain only data, while buffer status is maintained in the descriptor. Data chaining refers to data that spans multiple data buffers. However, a single

descriptor cannot span multiple data.
A single descriptor is used for both reception and transmission. The base address of the list is written into Descriptor List Base Address Register (SDMMC_DBADDR @0x88). A descriptor list is forward linked. The Last Descriptor can point back to the first entry in order to create a ring structure. The descriptor list resides in the physical memory address space of the Host. Each descriptor can point to a maximum of two data buffers.

1. IDMAC CSR Access

When an IDMAC is introduced, an additional CSR space resides in the IDMAC that controls the IDMAC functionality. The host accesses the new CSR space in addition to the existing control register set in the BIU. The IDMAC CSR primarily contains descriptor information. For a write operation to the CSR, the respective CSR logic of the IDMAC and BIU decodes the address before accepting. For a read operation from the CSR, the appropriate CSR read path is enabled.
You can enable or disable the IDMAC operation by programming bit[25] in the SDMMC_CTRL register of the BIU. This allows the data transfer by accessing the slave interface on the AMBA bus if the IDMAC is present but disabled. When IDMAC is enabled, the FIFO cannot be accessed through the slave interface.

2. Descriptors

- Descriptor structures

The IDMAC uses these types of descriptor structures:

- Dual-Buffer Structure – The distance between two descriptors is determined by the Skip Length value programmed in the Descriptor Skip Length (DSL) field of the Bus Mode Register (SDMMC_BMOD @0x80).

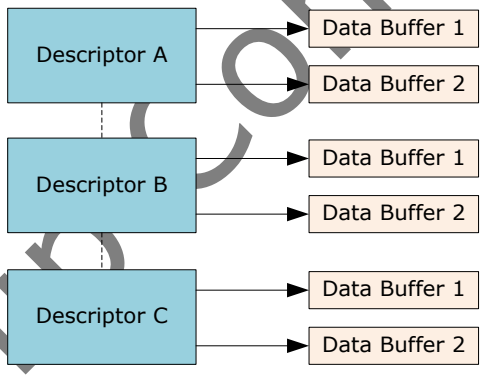

Fig. 7-6 Dual-Buffer Descriptor Structure

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

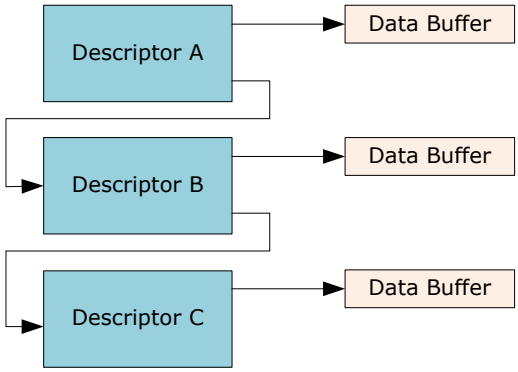

Fig. 7-7 Chain Descriptor Structure

- Descriptor formats

Following figure illustrates the internal formats of a descriptor. The descriptor addresses must be aligned to the bus width used for 32-bit AHB data buses. Each descriptor contains 16 bytes of control and status information. DES0 is a notation used to denote the [31:0] bits, DES1 to denote [63:32] bits, DES2 to denote [95:64] bits, DES3 to denote [127:96] bits.

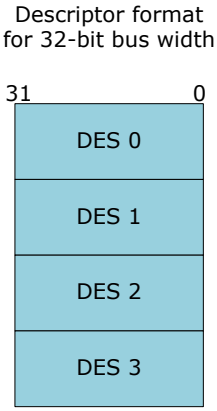

Fig. 7-8 Descriptor Formats for 32-bit AHB Address Bus Width

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

Table 7-4 Bits in IDMAC DES0 Element

Bit	Name	Description
31	OWN	When set, this bit indicates that the descriptor is owned by the IDMAC. When this bit is reset, it indicates that the descriptor is owned by the Host. The IDMAC clears this bit when it completes the data transfer.
30	Card Error Summary (CES)	These error bits indicate the status of the transaction to or from the card. These bits are also present in SDMMC_RINTSTS Indicates the logical OR of the following bits: ● EBE: End Bit Error ● RTO: Response Time out ● RCRC: Response CRC ● SBE: Start Bit Error ● DRTO: Data Read Timeout ● DCRC: Data CRC for Receive ● RE: Response Error
29:6	Reserved	-
5	End of Ring (ER)	When set, this bit indicates that the descriptor list reached its final descriptor. The IDMAC returns to the base address of the list, creating a Descriptor Ring. This is meaningful for only a dual-buffer descriptor structure.
4	Second Address Chained (CH)	When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is set, BS2 (DES1[25:13]) should be all zeros.
3	First Descriptor (FS)	When set, this bit indicates that this descriptor contains the first buffer of the data. If the size of the first buffer is 0, next Descriptor contains the beginning of the data.
2	Last Descriptor (LD)	This bit is associated with the last block of a DMA transfer. When set, the bit indicates that the buffers pointed to by this descriptor are the last buffers of the data. After this descriptor is completed, the remaining byte count is 0. In other words, after the descriptor with the LD bit set is completed, the remaining byte count should be 0.
1	Disable Interrupt on	When set, this bit will prevent the setting of the TI/RI bit of the IDMAC Status Register (IDSTS) for the data that ends in

Bit	Name	Description
	Completion (DIC)	the buffer pointed to by this descriptor.
0	Reserved	-

- The DES1 element contains the buffer size.

Table 7-5 Bits in IDMAC DES1 Element

Bit	Name	Description
31:26	Reserved	-
25:13	Buffer 2 Size (BS2)	These bits indicate the second data buffer byte size. The buffer size must be a multiple of 2, 4, or 8, depending upon the bus widths—16, 32, and 64 respectively. In the case where the buffer size is not a multiple of 2, 4, or 8, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and proceeds to the next buffer in case of a dual-buffer structure. This field is not valid for chain structure; that is, if DES0[4] is set.
12:0	Buffer 1 Size (BS1)	Indicates the data buffer byte size, which must be a multiple of 2, 4, or 8 bytes, depending upon the bus widths—16, 32, and 64, respectively. In the case where the buffer size is not a multiple of 2, 4, or 8, the resulting behavior is undefined. This field should not be zero. Note: If there is only one buffer to be programmed, you need to use only the Buffer 1, and not Buffer 2.

- The DES2 element contains the address pointer to the data buffer.

Table 7-6 Bits in IDMAC DES2 Element

Bit	Name	Description
31:26	Reserved	
25:13	Buffer 2 Size (BS2)	These bits indicate the second data buffer byte size. The buffer size must be a multiple of 2, 4, or 8, depending upon the bus widths—16, 32, and 64 respectively. In the case where the buffer size is not a multiple of 2, 4, or 8, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and proceeds to the next buffer in case of a dual-buffer structure. This field is not valid for chain structure; that is, if DES0[4] is set.
12:0	Buffer 1 Size (BS1)	Indicates the data buffer byte size, which must be a multiple of 2, 4, or 8 bytes, depending upon the bus widths—16, 32, and 64, respectively. In the case where the buffer size is not a multiple of 2, 4, or 8, the resulting behavior is undefined. This field should not be zero. Note: If there is only one buffer to be programmed, you need to use only the Buffer 1, and not Buffer 2.

- The DES3 element contains the address pointer to the next descriptor if the present descriptor is not the last descriptor in a chained descriptor structure or the second buffer address for a dual-buffer structure.

Table 7-7 Bits in IDMAC DES3 Element

Bit	Name	Description
31:0	Buffer Address Pointer 2/ Next Descriptor	These bits indicate the physical address of the second buffer when the dual-buffer structure is used. If the Second Address Chained (DES0[4]) bit is set, then this address contains the pointer to the physical memory where the Next

Bit	Name	Description
	Address (BAP2)	Descriptor is present. If this is not the last descriptor, then the Next Descriptor address pointer must be bus-width aligned.

3. Initialization

IDMAC initialization occurs as follows:

1) Write to IDMAC Bus Mode Register—SDMMC_BMOD to set Host bus access parameters.
2) Write to IDMAC Interrupt Enable Register—SDMMC_IDINTEN to mask unnecessary interrupt causes.
3) The software driver creates either the Transmit or the Receive descriptor list. Then it writes to IDMAC Descriptor List Base Address Register (SDMMC_DBADDR), providing the IDMAC with the starting address of the list.
4) The IDMAC engine attempts to acquire descriptors from the descriptor lists.

- Host Bus Burst Access

The IDMAC attempts to execute fixed-length burst transfers on the AHB Master interface if configured using the FB bit of the IDMAC Bus Mode register. The maximum burst length is indicated and limited by the PBL field. The descriptors are always accessed in the maximum possible burst-size for the 16-bytes to be read— 16*8/bus-width.
The IDMAC initiates a data transfer only when sufficient space to accommodate the configured burst is available in the FIFO or the number of bytes to the end of data, when less than the configured burst-length.
The IDMAC indicates the start address and the number of transfers required to the AHB Master Interface. When the AHB Interface is configured for fixed-length bursts, then it transfers data using the best combination of INCR4/8/16 and SINGLE transactions. Otherwise, in no fixed-length bursts, it transfers data using INCR (undefined length) and SINGLE transactions.

- Host Data Buffer Alignment

The Transmit and Receive data buffers in host memory must be aligned, depending on the data width.

- Buffer Size Calculations

The driver knows the amount of data to transmit or receive. For transmitting to the card, the IDMAC transfers the exact number of bytes to the FIFO, indicated by the buffer size field of DES1.
If a descriptor is not marked as last-LS bit of DES0-then the corresponding buffer(s) of the descriptor are full, and the amount of valid data in a buffer is accurately indicated by its buffer size field. If a descriptor is marked as last, then the buffer cannot be full, as indicated by the buffer size in DES1. The driver is aware of the number of locations that are valid in this case.

- Transmission

IDMAC transmission occurs as follows:

1) The Host sets up the elements (DES0-DES3) for transmission and sets the OWN bit (DES0[31]). The Host also prepares the data buffer.
2) The Host programs the write data command in the SDMMC_CMD register in BIU.
3) The Host will also program the required transmit threshold level (TX_WMark field in SDMMC_FIFOTH register).
4) The IDMAC determines that a write data transfer needs to be done as a consequence of step 2.
5) The IDMAC engine fetches the descriptor and checks the OWN bit. If the OWN bit is not set, it means that the host owns the descriptor. In this case the IDMAC enters suspend state and asserts the Descriptor Unable interrupt in the SDMMC_IDSTS register. In such

a case, the host needs to release the IDMAC by writing any value to the poll demand register.

6) It will then wait for Command Done (CD) bit and no errors from BIU which indicates that a transfer can be done.
7) The IDMAC engine will now wait for a DMA interface request from BIU. This request will be generated based on the programmed transmit threshold value. For the last bytes of data which can't be accessed using a burst, SINGLE transfers are performed on AHB Master Interface.
8) The IDMAC fetches the Transmit data from the data buffer in the Host memory and transfers to the FIFO for transmission to card.
9) When data spans across multiple descriptors, the IDMAC will fetch the next descriptor and continue with its operation with the next descriptor. The Last Descriptor bit in the descriptor indicates whether the data spans multiple descriptors or not.
10) When data transmission is complete, status information is updated in SDMMC_IDSTS register by setting Transmit Interrupt, if enabled. Also, the OWN bit is cleared by the IDMAC by performing a write transaction to DES0.

- Reception

IDMAC reception occurs as follows:

1) The Host sets up the element (DES0-DES3) for reception, sets the OWN (DES0[31]).
2) The Host programs the read data command in the SDMMC_CMD register in BIU.
3) The Host will program the required receive threshold level (RX_WMark field in FIFOTH register).
4) The IDMAC determines that a read data transfer needs to be done as a consequence of step 2.
5) The IDMAC engine fetches the descriptor and checks the OWN bit. If the OWN bit is not set, it means that the host owns the descriptor. In this case the DMA enters suspend state and asserts the Descriptor Unable interrupt in the SDMMC_IDSTS register. In such a case, the host needs to release the IDMAC by writing any value to the poll demand register.
6) It will then wait for Command Done (CD) bit and no errors from BIU which indicates that a transfer can be done.
7) The IDMAC engine will now wait for a DMA interface request from BIU. This request will be generated based on the programmed receive threshold value. For the last bytes of data which can't be accessed using a burst, SINGLE transfers are performed on AHB.
8) The IDMAC fetches the data from the FIFO and transfer to Host memory.
9) When data spans across multiple descriptors, the IDMAC will fetch the next descriptor and continue with its operation with the next descriptor. The Last Descriptor bit in the descriptor indicates whether the data spans multiple descriptors or not.
10) When data reception is complete, status information is updated in SDMMC_IDSTS register by setting Receive Interrupt, if enabled. Also, the OWN bit is cleared by the IDMAC by performing a write transaction to DES0.

- Interrupts

Interrupts can be generated as a result of various events. SDMMC_IDSTS register contains all the bits that might cause an interrupt. SDMMC_IDINTEN register contains an Enable bit for each of the events that can cause an interrupt.

There are two groups of summary interrupts-Normal and Abnormal-as outlined in SDMMC_IDSTS register. Interrupts are cleared by writing a 1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding

summary bit is cleared. When both the summary bits are cleared, the interrupt signal dmac_intr_o is de-asserted.
Interrupts are not queued and if the interrupt event occurs before the driver has responded to it, no additional interrupts are generated. For example, Receive Interrupt—SDMMC_IDSTS[1] indicates that one or more data was transferred to the Host buffer.
An interrupt is generated only once for simultaneous, multiple events. The driver must scan SDMMC_IDSTS register for the interrupt cause.

7.4 Register Description

7.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SDMMC_CTRL	0x0000	W	0x00000000	Control register
SDMMC_PWREN	0x0004	W	0x00000000	Power-enable register
SDMMC_CLKDIV	0x0008	W	0x00000000	Clock-divider register
SDMMC_CLKSRC	0x000c	W	0x00000000	SD Clock Source Register
SDMMC_CLKENA	0x0010	W	0x00000000	Clock-enable register
SDMMC_TMOUT	0x0014	W	0xffffff40	Time-out register
SDMMC_CTYPE	0x0018	W	0x00000000	Card-type register
SDMMC_BLKSIZ	0x001c	W	0x00000200	Block-size register
SDMMC_BYTCNT	0x0020	W	0x00000200	Byte-count register
SDMMC_INTMASK	0x0024	W	0x00000000	Interrupt-mask register
SDMMC_CMDARG	0x0028	W	0x00000000	Command-argument register
SDMMC_CMD	0x002c	W	0x00000000	Command register
SDMMC_RESP0	0x0030	W	0x00000000	Response-0 register
SDMMC_RESP1	0x0034	W	0x00000000	Response-1 register
SDMMC_RESP2	0x0038	W	0x00000000	Response-2 register
SDMMC_RESP3	0x003c	W	0x00000000	Response-3 register
SDMMC_MINTSTS	0x0040	W	0x00000000	Masked interrupt-status register
SDMMC_RINTSTS	0x0044	W	0x00000000	Raw interrupt-status register
SDMMC_STATUS	0x0048	W	0x00000406	Status register
SDMMC_FIFOTH	0x004c	W	0x00000000	FIFO threshold register
SDMMC_CDETECT	0x0050	W	0x00000000	Card-detect register
SDMMC_WRTPRT	0x0054	W	0x00000000	Write-protect register
SDMMC_TCBCNT	0x005c	W	0x00000000	Transferred CIU card byte count
SDMMC_TBBCNT	0x0060	W	0x00000000	Transferred host/DMA to/from BIU-FIFO byte count
SDMMC_DEBNCE	0x0064	W	0x00ffffff	Card detect debounce register
SDMMC_USRID	0x0068	W	0x07967797	User ID register
SDMMC_VERID	0x006c	W	0x5342270a	Synopsys version ID register
SDMMC_HCON	0x0070	W	0x00000000	Hardware configuration register
SDMMC_UHS_REG	0x0074	W	0x00000000	UHS-1 register
SDMMC_RSTN	0x0078	W	0x00000001	Hardware reset register
SDMMC_BMOD	0x0080	W	0x00000000	Bus mode register
SDMMC_PLDMND	0x0084	W	0x00000000	Poll demand register

Name	Offset	Size	Reset Value	Description
SDMMC_DBADDR	0x0088	W	0x00000000	Descriptor list base address register
SDMMC_IDSTS	0x008c	W	0x00000000	Internal DMAC status register
SDMMC_IDINTEN	0x0090	W	0x00000000	Internal DMAC interrupt enable register
SDMMC_DSCADDR	0x0094	W	0x00000000	Current host descriptor address register
SDMMC_BUFADDR	0x0098	W	0x00000000	Current buffer descriptor address register
SDMMC_CARDTHRCTL	0x0100	W	0x00000000	Card read threshold enable register
SDMMC_BACK_END_POWER	0x0104	W	0x00000000	Back-end power register
SDMMC_EMMC_DDR_REG	0x010c	W	0x00000000	eMMC 4.5 ddr start bit detection control register
SDMMC_FIFO_BASE	0x0200	W	0x00000000	FIFO base address register

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

7.4.2 Detail Register Description

SDMMC_CTRL

Address: Operational Base + offset (0x0000)

Bit	Attr	Reset Value	Description
31:26	RO	0x0	reserved
25	RW	0x0	use_internal_dmac Present only for the Internal DMAC configuration; else, it is reserved. 1'b0: The host performs data transfers through the slave interface 1'b1: Internal DMAC used for data transfe
24:12	RO	0x0	reserved
11	RW	0x0	ceata_device_interrupt_status 1'b0: Interrupts not enabled in CE-ATA device 1'b1: Interrupts are enabled in CE-ATA device Software should appropriately write to this bit after power-on reset or any other reset to CE-ATA device. After reset, usually CE-ATA device interrupt is disabled. If the host enables CE-ATA device interrupt, then software should set this bit.

Bit	Attr	Reset Value	Description
10	RW	0x0	send_auto_stop_ccsd 1'b0: Clear bit if the Host Controller does not reset the bit. 1'b1: Send internally generated STOP after sending CCSD to CE-ATA device. NOTE: Always set send_auto_stop_ccsd and send_ccsd bits together send_auto_stop_ccsd should not be set independent of send_ccsd. When set, the Host Controller automatically sends internally-generated STOP command (CMD12) to CE-ATA device. After sending internally-generated STOP command, Auto Command Done (ACD) bit in SDMMC_RINTSTS is set and generates interrupt to host if Auto Command Done interrupt is not masked. After sending the CCSD, the Host Controller automatically clears send_auto_stop_ccsd bit.
9	RW	0x0	send_ccsd 1'b0: Clear bit if the Host Controller does not reset the bit 1'b1: Send Command Completion Signal Disable (CCSD) to CE-ATA device When set, the Host Controller sends CCSD to CE-ATA device. Software sets this bit only if current command is expecting CCS (that is, RW_BLK) and interrupts are enabled in CE-ATA device.Once the CCSD pattern is sent to device, the Host Controller automatically clears send_ccsd bit. It also sets Command Done (CD) bit in SDMMC_RINTSTS register and generates interrupt to host if Command Done interrupt is not masked. NOTE: Once send_ccsd bit is set, it takes two card clock cycles to drive the CCSD on the CMD line. Due to this, during the boundary conditions it may happen that CCSD is sent to the CE-ATA device, even if the device signalled CCS.
8	RW	0x0	abort_read_data 1'b0: no change 1'b1: after suspend command is issued during read-transfer, software polls card to find when suspend happened. Once suspend occurs, software sets bit to reset data state-machine, which is waiting for next block of data. Bit automatically clears once data state machine resets to idle. Used in SDIO card suspend sequence.

Bit	Attr	Reset Value	Description
7	RW	0x0	send_irq_response 1'b0: no change 1'b1: send auto IRQ response Bit automatically clears once response is sent. To wait for MMC card interrupts, host issues CMD40, and SDMMC Controller waits for interrupt response from MMC card(s). In meantime, if host wants SDMMC Controller to exit waiting for interrupt state, it can set this bit, at which time SDMMC Controller command state-machine sends CMD40 response on bus and returns to idle state.
6	RW	0x0	read_wait 1'b0: clear read wait 1'b1: assert read wait For sending read-wait to SDIO cards.
5	RW	0x0	dma_enable 1'b0: disable DMA transfer mode 1'b1: enable DMA transfer mode Even when DMA mode is enabled, host can still push/pop data into or from FIFO; this should not happen during the normal operation. If there is simultaneous FIFO access from host/DMA, the data coherency is lost. Also, there is no arbitration inside SDMMC Controller to prioritize simultaneous host/DMA access.
4	RW	0x0	int_enable Global interrupt enable/disable bit: 1'b0: disable interrupts 1'b1: enable interrupts The int port is 1 only when this bit is 1 and one or more unmasked interrupts are set.
3	RO	0x0	reserved
2	RW	0x0	dma_reset 1'b0: no change 1'b1: reset internal DMA interface control logic To reset DMA interface, firmware should set bit to 1. This bit is auto-cleared after two AHB clocks.
1	RW	0x0	fifo_reset 1'b0: no change 1'b1: reset to data FIFO To reset FIFO pointers To reset FIFO, firmware should set bit to 1. This bit is auto-cleared after completion of reset operation.

Bit	Attr	Reset Value	Description
0	RW	0x0	controller_reset 1'b0: no change 1'b1: reset SDMMC controller To reset controller, firmware should set bit to 1. This bit is auto-cleared after two AHB and two cclk_in clock cycles. This resets: a. BIU/CIU interface b. CIU and state machines c. abort_read_data, send_irq_response, and read_wait bits of Control register d. start_cmd bit of Command register Does not affect any registers or DMA interface, or FIFO or host interrupts.

SDMMC_PWREN

Address: Operational Base + offset (0x0004)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	power_enable Power on/off switch for the card. Once power is turned on, firmware should wait for regulator/switch ramp-up time before trying to initialize card. 1'b0: power off 1'b1: power on Bit values output to card_power_en port.

SDMMC_CLKDIV

Address: Operational Base + offset (0x0008)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	clk_divider0 Clock divider-0 value. Clock division is 2*n. For example, value of 0 means divide by 2*0 = 0 (no division, bypass), value of 1 means divide by 2*1 = 2, and so on.

SDMMC_CLKSRC

Address: Operational Base + offset (0x000c)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved

Bit	Attr	Reset Value	Description
1:0	RW	0x0	clk_source Clock divider source for up to 16 SD cards supported. Each card has two bits assigned to it. For example, bits[1:0] assigned for card-0, which maps and internally routes clock divider[3:0] outputs to cclk_out[15:0] pins, depending on bit value. 2'b00: clock divider 0.

SDMMC_CLKENA

Address: Operational Base + offset (0x0010)

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved
16	RW	0x0	cclk_low_power Low-power control for SD card clock and MMC card clock supported. 1'b0: non-low-power mode 1'b1: low-power mode; stop clock when card in IDLE (should be normally set to only MMC and SD memory cards; for SDIO cards, if interrupts must be detected, clock should not be stopped.)
15:1	RO	0x0	reserved
0	RW	0x0	cclk_enable Clock-enable control for SD card clock and MMC card clock supported. 1'b0: clock disabled 1'b1: clock enabled

SDMMC_TMOUT

Address: Operational Base + offset (0x0014)

Bit	Attr	Reset Value	Description
31:8	RW	0x000000	data_timeout Value for card Data Read Timeout; same value also used for Data Starvation by Host timeout. Value is in number of card output clocks cclk_out of selected card. Note: The software timer should be used if the timeout value is in the order of 100 ms. In this case, read data timeout interrupt needs to be disabled.
7:0	RW	0x40	response_timeout Response timeout value. Value is in number of card output clock.

SDMMC_CTYPE

Address: Operational Base + offset (0x0018)

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

Bit	Attr	Reset Value	Description
16	RW	0x0	card_width_8 Indicates if card is 8-bit. 1'b0: non 8-bit mode 1'b1: 8-bit mode
15:1	RO	0x0	reserved
0	RW	0x0	card_width Indicates if card is 1-bit or 4-bit. 1'b0: 1-bit mode 1'b1: 4-bit mode

SDMMC_BLKSIZ

Address: Operational Base + offset (0x001c)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RW	0x0000	block_size Block size

SDMMC_BYTCNT

Address: Operational Base + offset (0x0020)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	byte_count Number of bytes to be transferred; should be integer multiple of Block Size for block transfers. For undefined number of byte transfers, byte count should be set to 0. When byte count is set to 0, it is responsibility of host to explicitly send stop/abort command to terminate data transfer.

SDMMC_INTMASK

Address: Operational Base + offset (0x0024)

Bit	Attr	Reset Value	Description
31:25	RO	0x0	reserved
24	RW	0x0	sdio_int_mask Mask SDIO interrupts. When masked, SDIO interrupt detection for that card is disabled. A 0 masks an interrupt, and 1 enables an interrupt.
23:17	RO	0x0	reserved
16	RW	0x0	data_nobusy_int_mask 1'b0: data no busy interrupt not masked 1'b1: data no busy interrupt masked

Bit	Attr	Reset Value	Description
15:0	RW	0x0000	int_mask Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt. bit 15: End-bit error (read)/Write no CRC (EBE) bit 14: Auto command done (ACD) bit 13: Start-bit error (SBE) bit 12: Hardware locked write error (HLE) bit 11: FIFO underrun/overrun error (FRUN) bit 10: Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9: Data read timeout (DRTO) bit 8: Response timeout (RTO) bit 7: Data CRC error (DCRC) bit 6: Response CRC error (RCRC) bit 5: Receive FIFO data request (RXDR) bit 4: Transmit FIFO data request (TXDR) bit 3: Data transfer over (DTO) bit 2: Command done (CD) bit 1: Response error (RE) bit 0: Card detect (CD)

SDMMC_CMDARG

Address: Operational Base + offset (0x0028)

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

SDMMC_CMD

Address: Operational Base + offset (0x002c)

Bit	Attr	Reset Value	Description
31	RW	0x0	start_cmd Start command. Once command is taken by CIU, bit is cleared. When bit is set, host should not attempt to write to any command registers. If write is attempted, hardware lock error is set in raw interrupt register. Once command is sent and response is received from SD_MMC cards, Command Done bit is set in raw interrupt register.
30	RO	0x0	reserved

Bit	Attr	Reset Value	Description
29	RW	0x0	use_hold_reg Use Hold Register. 1'b0: CMD and DATA sent to card bypassing HOLD Register 1'b1: CMD and DATA sent to card through the HOLD Register Note: a. Set to 1'b1 for SDR12 and SDR25 (with non-zero phase-shifted cclk_in_drv); zero phase shift is not allowed in these modes. b. Set to 1'b0 for SDR50, SDR104, and DDR50 (with zero phase-shifted cclk_in_drv). c. Set to 1'b1 for SDR50, SDR104, and DDR50 (with non-zero phase-shifted cclk_in_drv).
28	RW	0x0	volt_switch Voltage switch bit. 1'b0: no voltage switching 1'b1: voltage switching enabled; must be set for CMD11 only
27	RW	0x0	boot_mode Boot Mode. 1'b0: mandatory boot operation 1'b1: alternate boot operation
26	RW	0x0	disable_boot Disable boot. When software sets this bit along with start_cmd, CIU terminates the boot operation. Do NOT set disable_boot and enable_boot together.
25	RW	0x0	expect_boot_ack Expect Boot Acknowledge. When Software sets this bit along with enable_boot, CIU expects a boot acknowledge start pattern of 0-1-0 from the selected card.
24	RW	0x0	enable_boot Enable Boot. This bit should be set only for mandatory boot mode.When Software sets this bit along with start_cmd, CIU starts the boot sequence for the corresponding card by asserting the CMD line low. Do NOT set disable_boot and enable_boot together.
23	RW	0x0	ccs_expected 1'b0: command does not expect CCS from device 1'b1: command expects CCS from device If the command expects Command Completion Signal (CCS) from the CE-ATA device, the software should set this control bit. The Host Controller sets Data Transfer Over (DTO) bit in SDMMC_RINTSTS register and generates interrupt to host if Data Transfer Over interrupt is not masked.

Bit	Attr	Reset Value	Description
22	RW	0x0	read_ceata_device 1'b0: Host is not performing read access (RW_REG or RW_BLK) towards CE-ATA device 1'b1: Host is performing read access (RW_REG or RW_BLK) towards CE-ATA device Software should set this bit to indicate that CE-ATA device is being accessed for read transfer. This bit is used to disable read data timeout indication while performing CE-ATA read transfers. Maximum value of I/O transmission delay can be no less than 10 seconds. The Host Controller should not indicate read data timeout while waiting for data from CE-ATA device.
21	RW	0x0	update_clock_registers_only 1'b0: normal command sequence 1'b1: do not send commands, just update clock register value into card clock domain Following register values transferred into card clock domain: CLKDIV, CLRSRC, CLKENA. Changes card clocks (change frequency, truncate off or on, and set low-frequency mode); provided in order to change clock frequency or stop clock without having to send command to cards. During normal command sequence, when update_clock_registers_only=0, following control registers are transferred from BIU to CIU: CMD, CMDARG, TMOUT, CTYPE, BLKSIZ, BYTCNT. CIU uses new register values for new command sequence to card. When bit is set, there are no Command Done interrupts because no command is sent to SD_MMC_CEATA cards.
20:16	RO	0x0	reserved
15	RW	0x0	send_initialization 1'b0: do not send initialization sequence (80 clocks of 1) before sending this command 1'b1: send initialization sequence before sending this command After power on, 80 clocks must be sent to card for initialization before sending any commands to card. Bit should be set while sending first command to card so that controller will initialize clocks before sending command to card. This bit should not be set for either of the boot modes (alternate or mandatory).

Bit	Attr	Reset Value	Description
14	RW	0x0	stop_abort_cmd 1'b0: neither stop nor abort command to stop current data transfer in progress. If abort is sent to function-number currently selected or not in data-transfer mode, then bit should be set to 0. 1'b1: stop or abort command intended to stop current data transfer in progress. When open-ended or predefined data transfer is in progress, and host issues stop or abort command to stop data transfer, bit should be set so that command/data state-machines of CIU can return correctly to idle state. This is also applicable for Boot mode transfers. To Abort boot mode, this bit should be set along with CMD[26]=disable_boot.
13	RW	0x0	wait_prvdata_complete 1'b0: send command at once, even if previous data transfer has not completed 1'b1: wait for previous data transfer completion before sending command The wait_prvdata_complete=0 option typically used to query status of card during data transfer or to stop current data transfer; card_number should be same as in previous command.
12	RW	0x0	send_auto_stop 1'b0: no stop command sent at end of data transfer 1'b1: send stop command at end of data transfer When set, SDMMC Controller sends stop command to SD_MMC cards at end of data transfer. a. when send_auto_stop bit should be set, since some data transfers do not need explicit stop commands b. open-ended transfers that software should explicitly send to stop command Additionally, when "resume" is sent to resume-suspended memory access of SD-Combo card -bit should be set correctly if suspended data transfer needs send_auto_stop. Don't care if no data expected from card.
11	RW	0x0	transfer_mode 1'b0: block data transfer command 1'b1: stream data transfer command Don't care if no data expected.
10	RW	0x0	wr 1'b0: read from card 1'b1: write to card Don't care if no data expected from card.
9	RW	0x0	data_expected 1'b0: no data transfer expected (read/write) 1'b1: data transfer expected (read/write)

Bit	Attr	Reset Value	Description
8	RW	0x0	check_response_crc 1'b0: do not check response CRC 1'b1: check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.
7	RW	0x0	response_length 1'b0: short response expected from card 1'b1: long response expected from card
6	RW	0x0	response_expect 1'b0: no response expected from card 1'b1: response expected from card
5:0	RW	0x00	cmd_index Command index

SDMMC_RESP0

Address: Operational Base + offset (0x0030)

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

SDMMC_RESP1

Address: Operational Base + offset (0x0034)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	response1 Register represents bit[63:32] of long response. When CIU sends auto-stop command, then response is saved in register. Response for previous command sent by host is still preserved in Response 0 register. Additional auto-stop issued only for data transfer commands, and response type is always "short" for them.

SDMMC_RESP2

Address: Operational Base + offset (0x0038)

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

SDMMC_RESP3

Address: Operational Base + offset (0x003c)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	response3 Bit[127:96] of long response.

SDMMC_MINTSTS

Address: Operational Base + offset (0x0040)

Bit	Attr	Reset Value	Description
31:25	RO	0x0	reserved
24	RW	0x0	sdio_interrupt Interrupt from SDIO card; SDIO interrupt for card enabled only if corresponding sdio_int_mask bit is set in Interrupt mask register (mask bit 1 enables interrupt; 0 masks interrupt). 1'b0: no SDIO interrupt from card 1'b1: SDIO interrupt from card
23:17	RO	0x0	reserved
16	RW	0x0	data_nobusy_int_status Data no busy interrupt status, high active.
15:0	RW	0x0000	int_status Interrupt enabled only if corresponding bit in interrupt mask register is set. bit 15: End-bit error (read)/Write no CRC (EBE) bit 14: Auto command done (ACD) bit 13: Start-bit error (SBE) bit 12: Hardware locked write error (HLE) bit 11: FIFO underrun/overrun error (FRUN) bit 10: Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9: Data read timeout (DRTO) bit 8: Response timeout (RTO) bit 7: Data CRC error (DCRC) bit 6: Response CRC error (RCRC) bit 5: Receive FIFO data request (RXDR) bit 4: Transmit FIFO data request (TXDR) bit 3: Data transfer over (DTO) bit 2: Command done (CD) bit 1: Response error (RE) bit 0: Card detect (CD)

SDMMC_RINTSTS

Address: Operational Base + offset (0x0044)

Bit	Attr	Reset Value	Description
31:25	RO	0x0	reserved
24	RW	0x0	sdio_interrupt Interrupt from SDIO card; Writes to these bits clear them. Value of 1 clears bit and 0 leaves bit intact. 1'b0: no SDIO interrupt from card 1'b1: SDIO interrupt from card
23:17	RO	0x0	reserved
16	RW	0x0	data_nobusy_int_status Data no busy interrupt status, high active.

Bit	Attr	Reset Value	Description
15:0	RW	0x0000	int_status Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. bit 15: End-bit error (read)/Write no CRC (EBE) bit 14: Auto command done (ACD) bit 13: Start-bit error (SBE) bit 12: Hardware locked write error (HLE) bit 11: FIFO underrun/overrun error (FRUN) bit 10: Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9: Data read timeout (DRTO) bit 8: Response timeout (RTO) bit 7: Data CRC error (DCRC) bit 6: Response CRC error (RCRC) bit 5: Receive FIFO data request (RXDR) bit 4: Transmit FIFO data request (TXDR) bit 3: Data transfer over (DTO) bit 2: Command done (CD) bit 1: Response error (RE) bit 0: Card detect (CD)

SDMMC_STATUS

Address: Operational Base + offset (0x0048)

Bit	Attr	Reset Value	Description
31	RW	0x0	dma_req DMA request signal state.
30	RO	0x0	dma_ack DMA acknowledge signal state.
29:17	RO	0x0000	fifo_count Number of filled locations in FIFO.
16:11	RO	0x00	response_index Index of previous response, including any auto-stop sent by core.
10	RO	0x1	data_state_mc_busy Data transmit or receive state-machine is busy.
9	RW	0x0	data_busy Inverted version of raw selected card_data[0]. 1'b0: card data not busy 1'b1: card data busy default value is 1 or 0 depending on cdata_in.
8	RW	0x0	data_3_status Raw selected card_data[3]; checks whether card is present. 1'b0: card not present 1'b1: card present default value is 1 or 0 depending on cdata_in.

Bit	Attr	Reset Value	Description
7:4	RW	0x0	command_fsm_states Command FSM states: 4'h0: idle 4'h1: send init sequence 4'h2: Tx cmd start bit 4'h3: Tx cmd tx bit 4'h4: Tx cmd index + arg 4'h5: Tx cmd crc7 4'h6: Tx cmd end bit 4'h7: Rx resp start bit 4'h8: Rx resp IRQ response 4'h9: Rx resp tx bit 4'ha: Rx resp cmd idx 4'hb: Rx resp data 4'hc: Rx resp crc7 4'hd: Rx resp end bit 4'he: Cmd path wait NCC 4'hf: Wait; CMD-to-response turnaround The command FSM state is represented using 19 bits. The SDMMC_STATUS Register[7:4] has 4 bits to represent the command FSM states. Using these 4 bits, only 16 states can be represented. Thus three states cannot be represented in the SDMMC_STATUS[7:4] register. The three states that are not represented in the SDMMC_STATUS Register[7:4] are: Bit 16: Wait for CCS Bit 17: Send CCSD Bit 18: Boot Mode Due to this, while command FSM is in "Wait for CCS state" or "Send CCSD" or "Boot Mode", the Status register indicates status as 0 for the bit field [7:4].
3	RO	0x0	fifo_full FIFO is full status.
2	RO	0x1	fifo_empty FIFO is empty status.
1	RO	0x1	fifo_tx_watermark FIFO reached Transmit watermark level; not qualified with data transfer.
0	RO	0x0	fifo_rx_watermark FIFO reached Receive watermark level; not qualified with data transfer.

SDMMC_FIFOTH

Address: Operational Base + offset (0x004c)

Bit	Attr	Reset Value	Description
31	RO	0x0	reserved

Bit	Attr	Reset Value	Description
30:28	RW	0x0	dma_mutiple_transaction_size Burst size of multiple transaction; should be programmed same as DMA controller multiple-transaction-size SRC/DEST_MSIZE. 3'b000: 1 transfers 3'b001: 4 3'b010: 8 3'b011: 16 3'b100: 32 3'b101: 64 3'b110: 128 3'b111: 256 The unit for transfer is the H_DATA_WIDTH parameter. A single transfer (dw_dma_single assertion in case of Non DW DMA interface) would be signalled based on this value. Value should be sub-multiple of (RX_WMark + 1)* (F_DATA_WIDTH/H_DATA_WIDTH) and (FIFO_DEPTH - TX_WMark)* (F_DATA_WIDTH/ H_DATA_WIDTH) For example, if FIFO_DEPTH = 16, F_DATA_WIDTH == H_DATA_WIDTH Allowed combinations for MSize and TX_WMark are: MSize = 1, TX_WMARK = 1-15 MSize = 4, TX_WMark = 8 MSize = 4, TX_WMark = 4 MSize = 4, TX_WMark = 12 MSize = 8, TX_WMark = 8 MSize = 8, TX_WMark = 4 Allowed combinations for MSize and RX_WMark are: MSize = 1, RX_WMARK = 0-14 MSize = 4, RX_WMark = 3 MSize = 4, RX_WMark = 7 MSize = 4, RX_WMark = 11 MSize = 8, RX_WMark = 7 Recommended: MSize = 8, TX_WMark = 8, RX_WMark = 7

Bit	Attr	Reset Value	Description
27:16	RW	0x000	rx_wmark FIFO threshold watermark level when receiving data to card. When FIFO data count reaches greater than this number, DMA/FIFO request is raised. During end of packet, request is generated regardless of threshold programming in order to complete any remaining data. In non-DMA mode, when receiver FIFO threshold (RXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, interrupt is not generated if threshold programming is larger than any remaining data. It is responsibility of host to read remaining bytes on seeing Data Transfer Done interrupt. In DMA mode, at end of packet, even if remaining bytes are less than threshold, DMA request does single transfers to flush out any remaining bytes before Data Transfer Done interrupt is set. 12 bits-1 bit less than FIFO-count of status register, which is 13 bits. Limitation: RX_WMark <= FIFO_DEPTH-2 Recommended: (FIFO_DEPTH/2) - 1; (means greater than (FIFO_DEPTH/2) - 1) NOTE: In DMA mode during CCS time-out, the DMA does not generate the request at the end of packet, even if remaining bytes are less than threshold. In this case, there will be some data left in the FIFO. It is the responsibility of the application to reset the FIFO after the CCS timeout.
15:12	RO	0x0	reserved
11:0	RW	0x000	tx_wmark FIFO threshold watermark level when transmitting data to card. When FIFO data count is less than or equal to this number, DMA/FIFO request is raised. If Interrupt is enabled, then interrupt occurs. During end of packet, request or interrupt is generated, regardless of threshold programming. In non-DMA mode, when transmit FIFO threshold (TXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, on last interrupt, host is responsible for filling FIFO with only required remaining bytes (not before FIFO is full or after CIU completes data transfers, because FIFO may not be empty). In DMA mode, at end of packet, if last transfer is less than burst size, DMA controller does single cycles until required bytes are transferred. 12 bits -1 bit less than FIFO-count of status register, which is 13 bits. Limitation: TX_WMark >= 1; Recommended: FIFO_DEPTH/2; (means less than or equal to FIFO_DEPTH/2).

SDMMC_CDETECT

Address: Operational Base + offset (0x0050)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	card_detect_n Value on card_detect_n input ports; read-only bits. 0 represents presence of card.

SDMMC_WRTPRT

Address: Operational Base + offset (0x0054)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	write_protect Value on card_write_prt input port. 1 represents write protection.

SDMMC_TCBCNT

Address: Operational Base + offset (0x005c)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	trans_card_byte_count Number of bytes transferred by CIU unit to card. Both SDMMC_TCBCNT and SDMMC_TBBCNT share same coherency register.

SDMMC_TBBCNT

Address: Operational Base + offset (0x0060)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	trans_fifo_byte_count Number of bytes transferred between Host/DMA memory and FIFO. Both SDMMC_TCBCNT and SDMMC_TBBCNT share same coherency register.

SDMMC_DEBNCE

Address: Operational Base + offset (0x0064)

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:0	RW	0xffffff	debounce_count Number of host clocks (clk) used by debounce filter logic; typical debounce time is 5-25 ms.

SDMMC_USRID

Address: Operational Base + offset (0x0068)

Bit	Attr	Reset Value	Description
31:0	RW	0x07967797	usrid User identification register; value set by user. Default reset value can be picked by user while configuring core before synthesis. Can also be used as scratch pad register by user. The default value is determined by Configuration Value.

SDMMC_VERID

Address: Operational Base + offset (0x006c)

Bit	Attr	Reset Value	Description
31:0	RO	0x5342270a	verid Version identification register; register value is hard-wired. Can be read by firmware to support different versions of core.

SDMMC_HCON

Address: Operational Base + offset (0x0070)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	HCON Configuration Dependent.Hardware configurations selected by user before synthesizing core. Register values can be used to develop configuration-independent software drivers. [0]: CARD_TYPE 1'b0: MMC_ONLY 1'b1: SD_MMC [5:1]: NUM_CARDS - 1 [6]: H_BUS_TYPE 1'b0: APB 1'b1: AHB [9:7]: H_DATA_WIDTH 3'b000: 16 bits 3'b001: 32 bits 3'b010: 64 bits others: reserved [15:10]: H_ADDR_WIDTH 0 to 7: reserved 6'd8: 9 bits 6'd9: 10 bits … 6'd31: 32 bits 6'd32 to 63: reserved [17:16]: DMA_INTERFACE 2'b00: none 2'b01: DW_DMA 2'b10: GENERIC_DMA 2'b11: NON-DW-DMA [20:18]: GE_DMA_DATA_WIDTH 3'b000: 16 bits 3'b001: 32 bits 3'b010: 64 bits others: reserved [21]: FIFO_RAM_INSIDE 1'b0: outside 1'b1: inside [22]: IMPLEMENT_HOLD_REG 1'b0: no hold register 1'b1: hold register [23]: SET_CLK_FALSE_PATH 1'b0: no false path 1'b1: false path set [25:24]: NUM_CLK_DIVIDER-1 [26]: AREA_OPTIMIZED 1'b0: no area optimization 1'b1: Area optimization

SDMMC_UHS_REG

Address: Operational Base + offset (0x0074)

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved
16	RW	0x0	ddr_reg DDR mode. 1'b0: non-DDR mode 1'b1: DDR mode
15:0	RO	0x0	reserved

SDMMC_RSTN

Address: Operational Base + offset (0x0078)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	card_reset Hardware reset. 1'b0: active mode 1'b1: reset

SDMMC_BMOD

Address: Operational Base + offset (0x0080)

Bit	Attr	Reset Value	Description
31:11	RO	0x0	reserved
10:8	RW	0x0	PBL Programmable Burst Length. These bits indicate the maximum number of beats to be performed in one IDMAC transaction. The IDMAC will always attempt to burst as specified in PBL each time it starts a Burst transfer on the host bus. The permissible values are 1, 4, 8, 16, 32, 64, 128 and 256. This value is the mirror of MSIZE of FIFOTH register. In order to change this value, write the required value to FIFOTH register. This is an encode value as follows. 3'b000: 1 transfers 3'b001: 4 transfers 3'b010: 8 transfers 3'b011: 16 transfers 3'b100: 32 transfers 3'b101: 64 transfers 3'b110: 128 transfers 3'b111: 256 transfers Transfer unit is either 16, 32, or 64 bits, based on HDATA_WIDTH. PBL is a read-only value and is applicable only for Data Access; it does not apply to descriptor accesses.
7	RW	0x0	DE IDMAC Enable. When set, the IDMAC is enabled.

Bit	Attr	Reset Value	Description
6:2	RW	0x00	DSL Descriptor Skip Length. Specifies the number of HWord/Word/Dword (depending on 16/32/64-bit bus) to skip between two unchained descriptors. This is pplicable only for dual buffer structure.
1	RW	0x0	FB Fixed Burst. Controls whether the AHB Master interface performs fixed burst transfers or not. When set, the AHB will use only SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers. When reset, the AHB will use SINGLE and INCR burst transfer operations.
0	RW	0x0	SWR Software Reset. When set, the DMA Controller resets all its internal registers. It is automatically cleared after 1 clock cycle.

SDMMC_PLDMND

Address: Operational Base + offset (0x0084)

Bit	Attr	Reset Value	Description
31:0	WO	0x00000000	PD Poll Demand. If the OWN bit of a descriptor is not set, the FSM goes to the Suspend state. The host needs to write any value into this register for the IDMAC FSM to resume normal descriptor fetch operation. This is a write only register.

SDMMC_DBADDR

Address: Operational Base + offset (0x0088)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	SDL Start of Descriptor List. Contains the base address of the First Descriptor. The LSB bits [0/1/2:0] for 16/32/64-bit bus-width) are ignored and taken as all-zero by the IDMAC internally. Hence these LSB bits are read-only.

SDMMC_IDSTS

Address: Operational Base + offset (0x008c)

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

Bit	Attr	Reset Value	Description
16:13	RW	0x0	FSM DMAC FSM present state. 4'h0: DMA_IDLE 4'h1: DMA_SUSPEND 4'h2: DESC_RD 4'h3: DESC_CHK 4'h4: DMA_RD_REQ_WAI 4'h5: DMA_WR_REQ_WAI 4'h6: DMA_RD 4'h7: DMA_WR 4'h8: DESC_CLOSE
12:10	RW	0x0	EB Error Bits. Indicates the type of error that caused a Bus Error. Valid only with fatal Bus. 3'h1: Host Abort received during transmission 3'h2: Host Abort received during reception Others: Reserved
9	RW	0x0	AIS Abnormal Interrupt Summary. Logical OR of the following: SDMMC_IDSTS[2] Fatal Bus Interrupt SDMMC_IDSTS[4] DU bit Interrupt Only unmasked bits affect this bit. This is a sticky bit and must be cleared each time a corresponding bit that causes AIS to be set is cleared. Writing a 1 clears this bit.
8	RW	0x0	NIS Normal Interrupt Summary. Logical OR of the following: SDMMC_IDSTS[0] Transmit Interrupt SDMMC_IDSTS[1] Receive Interrupt Only unmasked bits affect this bit. This is a sticky bit and must be cleared each time a corresponding bit that causes NIS to be set is cleared. Writing a 1 clears this bit.
7:6	RO	0x0	reserved

Bit	Attr	Reset Value	Description
5	RW	0x0	CES Card Error Summary. Indicates the status of the transaction to/from the card; also present in SDMMC_RINTSTS. Indicates the logical OR of the following bits: EBE: End Bit Error RTO: Response Timeout/Boot Ack Timeout RCRC: Response CRC SBE: Start Bit Error DRTO: Data Read Timeout/BDS timeout DCRC: Data CRC for Receive RE: Response Error Writing a 1 clears this bit. The abort condition of the IDMAC depends on the setting of this CES bit. If the CES bit is enabled, then the IDMAC aborts on a "response error"; however, it will not abort if the CES bit is cleared.
4	RW	0x0	DU Descriptor Unavailable Interrupt. This bit is set when the descriptor is unavailable due to OWN bit = 0 (DES0[31] =0). Writing a 1 clears this bit.
3	RO	0x0	reserved
2	RW	0x0	FBE Fatal Bus Error Interrupt. When this bit is set, the DMA disables all its bus accesses. Writing a 1 clears this bit.
1	RW	0x0	RI Receive Interrupt. Indicates the completion of data reception for a descriptor. Writing a 1 clears this bit.
0	RW	0x0	TI Transmit Interrupt. Indicates that data transmission is finished for a descriptor. Writing 1 clears this bit.

SDMMC_IDINTEN

Address: Operational Base + offset (0x0090)

Bit	Attr	Reset Value	Description
31:10	RO	0x0	reserved

Bit	Attr	Reset Value	Description
9	RW	0x0	AI Abnormal Interrupt Summary Enable. When set, an abnormal interrupt is enabled. This bit enables the following bits: SDMMC_IDINTEN[2] Fatal Bus Error Interrupt SDMMC_IDINTEN[4] DU Interrupt
8	RW	0x0	NI Normal Interrupt Summary Enable. When set, a normal interrupt is enabled. When reset, a normal interrupt is disabled. This bit enables the following bits: SDMMC_IDINTEN[0] Transmit Interrupt SDMMC_IDINTEN[1] Receive Interrup
7:6	RO	0x0	reserved
5	RW	0x0	CES Card Error summary Interrupt Enable. When set, it enables the Card Interrupt summary.
4	RW	0x0	DU Descriptor Unavailable Interrupt. When set along with Abnormal Interrupt Summary Enable, the DU interrupt is enabled.
3	RO	0x0	reserved
2	RW	0x0	FBE Fatal Bus Error Enable. When set with Abnormal Interrupt Summary Enable, the Fatal Bus Error Interrupt is enabled. When reset, Fatal Bus Error Enable Interrupt is disabled.
1	RW	0x0	RI Receive Interrupt Enable. When set with Normal Interrupt Summary Enable, Receive Interrupt is enabled. When reset, Receive Interrupt is disabled.
0	RW	0x0	TI Transmit Interrupt Enable. When set with Normal Interrupt Summary Enable, Transmit Interrupt is enabled. When reset, Transmit Interrupt is disabled.

SDMMC_DSCADDR

Address: Operational Base + offset (0x0094)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	HDA Host Descriptor Address Pointer. Cleared on reset. Pointer updated by IDMAC during operation. This register points to the start address of the current descriptor read by the IDMAC.

SDMMC_BUFADDR

Address: Operational Base + offset (0x0098)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	HBA Host Buffer Address Pointer. Cleared on Reset. Pointer updated by IDMAC during operation. This register points to the current Data Buffer Address being accessed by the IDMAC.

SDMMC_CARDTHRCTL

Address: Operational Base + offset (0x0100)

Bit	Attr	Reset Value	Description
31:28	RO	0x0	reserved
27:16	RW	0x000	CardRdThreshold Card Read Threshold size
15:2	RO	0x0	reserved
1	RW	0x0	BsyClrIntEn Busy Clear Interrupt generation. 1'b0: Busy Clear Interrupt disabled 1'b1: Busy Clear Interrupt enabled Note: The application can disable this feature if it does not want to wait for a Busy Clear Interrupt. For example, in a multi-card scenario, the application can switch to the other card without waiting for a busy to be completed. In such cases, the application can use the polling method to determine the status of busy. By default this feature is disabled and backward-compatible to the legacy drivers where polling is used.
0	RW	0x0	CardRdThrEn Card Read Threshold Enable. 1'b0: Card Read Threshold disabled 1'b1: Card Read Threshold enabled. Host Controller initiates Read Transfer only if CardRdThreshold amount of space is available in receive FIFO.

SDMMC_BACK_END_POWER

Address: Operational Base + offset (0x0104)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	back_end_power Back end power. 1'b0: Off; Reset 1'b1: Back-end Power supplied to card application

SDMMC_EMMC_DDR_REG

Address: Operational Base + offset (0x010c)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	HALF_START_BIT Control for start bit detection mechanism inside the Host Controller based on duration of start bit; each bit refers to one slot. For eMMC 4.5, start bit can be: 1'b0: Full cycle (HALF_START_BIT=0) 1'b1: Less than one full cycle (HALF_START_BIT=1) Set HALF_START_BIT=1 for eMMC 4.5 and above; set to 0 for SD applications.

SDMMC_FIFO_BASE

Address: Operational Base + offset (0x0200)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	fifo_base_addr Fifo base addr.

7.5 Interface Description

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

Table 7-8 SDMMC Interface Description

Module Pin	Direction	Pin Name	IOMUX Setting
sdmmc_cclk	O	SRADC0/PWM0_M0/UART1_CTSN_M0/SPI0_CS0n_M1/I2S_SDO1_M0/SDMMC_CLKOUT/TKEY9/GPIO0_C0_u	GRF_GPIO0C_IOMUX_L[2:0]=3'b110
sdmmc_ccmd	I/O	SRADC1/PWM1_M0/UART1_RTSN_M0/SPI0_CLK_M1/I2C1_SDA_M1/I2S_SCLK_RX_M0/SDMMC_CMD/TKEY10/GPIO0_C1_u	GRF_GPIO0C_IOMUX_L[7:4]=4'b0111
sdmmc_cdata0	I/O	SRADC2/PWM2_M0/UART1_RX_M0/SPI0_MOSI_M1/I2C1_SCL_M1/I2S_LRCK_RX_M0/SDMMC_D0/TKEY11/GPIO0_C2_u	GRF_GPIO0C_IOMUX_L[11:8]=4'b0111

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

7.6 Application Notes

As the interface description shows, SD/MMC has only one SDMMC_D0 IO, so the DDR mode introduced below can be ignored.Because of DDR mode only support 4bit and 8bit data line.

7.6.1 SD/MMC Termination Requirement

Following Figure illustrates the SD/MMC termination requirements, which is required to pull up ccmd and cdata lines on the device bus. The recommended specification for pull-up on the ccmd line (Rcmd) is 4.7K - 100K for MMC, and 10K - 100K for an SD. The recommended pull-up on the cdata line (Rdat) is 50K - 100K.

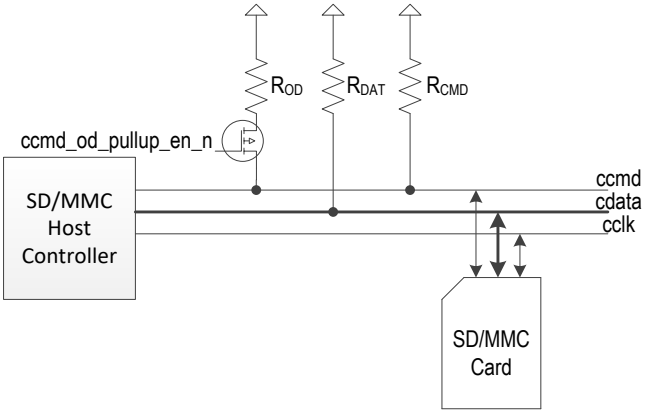

Fig. 7-9 SD/MMC Card Termination

Rcmd and Rod Calculation

The SD/MMC card enumeration happens at a very low frequency – 100-400KHz. Since the MMC bus is a shared bus between multiple cards, during enumeration open-drive mode is used to avoid bus conflict. Cards that drive 0 win over cards that drive "z". The pull-up in the command line pulls the bus to 1 when all cards drive "z". During normal data transfer, the host chooses only one card and the card driver switches to push-pull mode.

For example, if enumeration is done at 400KHz and the total bus capacitance is 200 pf, the pull-up needed during enumeration is:

2.2 RC = rise-time = 1/400KHz

R = 1/(2.2 * C * 100KHz)

= 1/(2.2 x 200 x 10**-12 x 400 x 10**3)

= 1/(17.6 x 10**-5)

= 5.68K

The ROD and RCMD should be adjusted in such a way that the effective pull-up is at the maximum 5.68K during enumeration. If there are only a few cards in the bus, a fixed RCMD resister is sufficient and there is no need for an additional ROD pull-up during enumeration. You should also ensure the effective pull-up will not violate the Iol rating of the drivers.

In SD mode, since each card has a separate bus, the capacitance is less, typically in the order of 20-30pf (host capacitance + card capacitance + trace + socket capacitance). For example, if enumeration is done at 400KHz and the total bus capacitance is 20pf, the pull-up needed during enumeration is:

2.2 RC = rise-time = 1/400KHz

R = 1/(2.2 * C * 100KHz)

= 1/(2.2 x 20 x 10**-12 x 400 x 10**3)

= 1/(1.76 x 10**-5)

= 56.8K

Therefore, a fixed 56.8K permanent Rcmd is sufficient in SD mode to enumerate the cards. The driver of the SD/MMC on the "command" port needs to be only a push-pull driver. During enumeration, the SD/MMC emulates an open-drain driver by driving only a 0 or a"z" by controlling the ccmd_out and ccmd_out_en signals.

7.6.2 Software/Hardware Restriction

Before issuing a new data transfer command, the software should ensure that the card is not busy due to any previous data transfer command. Before changing the card clock frequency, the software must ensure that there are no data or command transfers in progress.

If the card is enumerated in SDR50, or DDR50 mode, then the application must program

the use_hold_reg bit[29] in the SDMMC_CMD register to 1'b0 (phase shift of cclk_in_drv = 0) or 1'b1 (phase shift of cclk_in_drv>0). If the card is enumerated in SDR12 or SDR25 mode, the application must program the use_hold_reg bit[29] in the SDMMC_CMD register to 1'b1.

This programming should be done for all data transfer commands and non-data commands that are sent to the card. When the use_hold_reg bit is programmed to 1'b0, the Host Controller bypasses the Hold Registers in the transmit path. The value of this bit should not be changed when a Command or Data Transfer is in progress. For more details on using use_hold_reg and the implementation requirements for meeting the Card input hold time, refer to "Recommended Usage" in following table.

Table 7-9 Recommended Usage of use_hold_reg

No.	Speed Mode	use_hold_reg	cclk_in (MHz)	clk_in_drv (MHz)	clk_divider	Phase shift
1	SDR50	1'b0	100	100	0	0
2	SDR50	1'b1	100	100	0	Tunable> 0
3	SDR25	1'b1	50	50	0	Tunable> 0
4	SDR12	1'b1	50	50	1	Tunable> 0

To avoid glitches in the card clock outputs, the software should use the following steps when changing the card clock frequency:

1) Before disable the clocks, ensure that the card is not busy due to any previous data command. To determine this, check for 0 in bit9 of STATUS register.

2) Update the Clock Enable register to disable all clocks. To ensure completion of any previous command before this update, send a command to the CIU to update the clock registers by setting:

- start_cmd bit
- "update clock registers only" bits
- "wait_previous data complete" bit

Wait for the CIU to take the command by polling for 0 on the start_cmd bit.

3) Set the start_cmd bit to update the Clock Divider and/or Clock Source registers, and send a command to the CIU in order to update the clock registers; wait for the CIU to take the command.

4) Set start_cmd to update the Clock Enable register in order to enable the required clocks and send a command to the CIU to update the clock registers; wait for the CIU to take the command.

In non-DMA mode, while reading from a card, the Data Transfer Over (SDMMC_RINTSTS[3]) interrupt occurs as soon as the data transfer from the card is over. There still could be some data left in the FIFO, and the RX_WMark interrupt may or may not occur, depending on the remaining bytes in the FIFO. Software should read any remaining bytes upon seeing the Data Transfer Over (DTO) interrupt. While using the external DMA interface for reading from a card, the DTO interrupt occurs only after all the data is flushed to memory by the DMA interface unit.

While writing to a card in external DMA mode, if an undefined-length transfer is selected by setting the Byte Count Register to 0, the DMA logic will likely request more data than it will send to the card, since it has no way of knowing at which point the software will stop the transfer. The DMA request stops as soon as the DTO is set by the CIU.

If the software issues a controller_reset command by setting control register bit[0] to 1, all the CIU state machines are reset; the FIFO is not cleared. The DMA sends all remaining bytes to the host. In addition to a card-reset, if a FIFO reset is also issued, then:

- Any pending DMA transfer on the bus completes correctly
- DMA data read is ignored
- Write data is unknown(x)

Additionally, if dma_reset is also issued, any pending DMA transfer is abruptly terminated. When the DMA is used, the DMA controller channel should also be reset and reprogrammed.

If any of the previous data commands do not properly terminate, then the software should issue the FIFO reset in order to remove any residual data, if any, in the FIFO. After

asserting the FIFO reset, you should wait until this bit is cleared.
One data-transfer requirement between the FIFO and host is that the number of transfers should be a multiple of the FIFO data width (32bits). For example, you want to write only 15 bytes to an SD/MMC card (SDMMC_BYTCNT), the host should write 16 bytes to the FIFO or program the DMA to do 16-byte transfers. The software can still program the Byte Count register to only 15, at which point only 15 bytes will be transferred to the card. Similarly, when 15 bytes are read from a card, the host should still read all 16 bytes from the FIFO. It is recommended that you not change the FIFO threshold register in the middle of data transfers.

7.6.3 Programming Sequence

1. Initialization

Following figure illustrates the initialization flow.

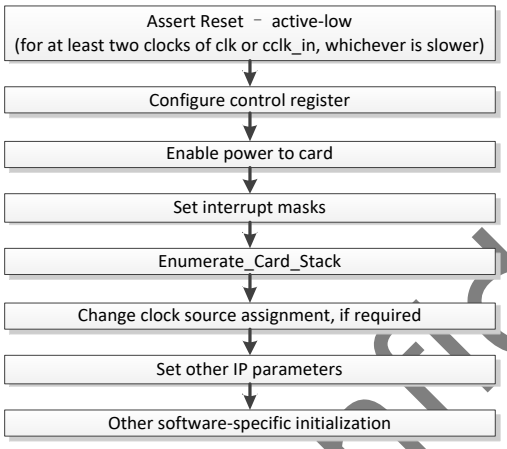

Fig. 7-10 Host Controller Initialization Sequence

Once the power and clocks are stable, reset_n should be asserted(active-low) for at least two clocks of clk or cclk_in, whichever is slower. The reset initializes the registers, ports, FIFO-pointers, DMA interface controls, and state-machines in the design. After power-on reset, the software should do the following:

1) Configure control register – For MMC mode, enable the open-drain pullup by setting enable_OD_pullup(bit24) in the control register.
2) Enable power to cards – Before enabling the power, confirm that the voltage setting to the voltage regulators is correct. Enable power to the connected cards by setting the corresponding bit to 1 in the Power Enable register. Wait for the power ramp-up time.
3) Set masks for interrupts by clearing appropriate bits in the Interrupt Mask register. Set the global int_enable bit of the Control register. It is recommended that you write 0xffff_ffff to the Raw Interrupt register in order to clear any pending interrupts before setting the int_enable bit.
4) Enumerate card stack – Each card is enumerated according to card type; for details, refer to "Enumerated Card Stack". For enumeration, you should restrict the clock frequency to 400KHz.
5) Changing clock source assignment – set the card frequency using the clock-divider and clock-source registers; for details, refer to "Clock Programming". MMC cards operate at a maximum of 20MHz (at maximum of 52MHz in high-speed mode). SD mode operates at a maximum of 25MHz (at maximum of 50MHz in high-speed mode).
6) Set other parameters, which normally do not need to be changed with every command, with a typical value such as timeout values in cclk_out according to SD/MMC specifications.

- ResponseTimeOut = 0x64
- DataTimeOut = highest of one of the following:
- (10*((TAAC*Fop)+(100*NSAC))

- Host FIFO read/write latency from FIFO empty/full
- Set the debounce value to 25ms(default:0x0fffff) in host clock cycle units in the DEBNCE register.
- FIFO threshold value in bytes in the SDMMC_FIFOTH register.

2. Enumerated Card Stack

The card stack does the following:

- Enumerates all connected cards
- Sets the RCA for the connected cards
- Reads card-specific information
- Stores card-specific information locally

Enumeration depends on the operating mode of the SD/MMC card; the card type is first identified and the appropriate card enumeration routine is called.

1) Check if the card is connected.
2) Clear the card type register to set the card width as a single bit. For the given card number, clear the corresponding bits in the card_type register. Clear the register bit for a 1-bit, 4-bit bus width. For example, for card number=1, clear bit 0 and bit 16 of the card_type register.
3) Set clock frequency to FOD=400KHz, maximum – Program clock divider0 (bits 0-7 in the CLKDIV register) value to one-half of the cclk_in frequency divided by 400KHz. For example, if cclk_in is 20MHz, then the value is 20, 000/(2*400)=25.
4) Identify the card type; that is, SD, MMC, or SDIO.
   a. Send CMD5 first. If a response is received, then the card is SDIO
   b. If not, send CMD8 with the following Argument
      Bit[31:12] = 20'h0 //reserved bits
      Bit[11:8] = 4'b0001 //VHS value
      Bit[7:0] = 8'b10101010 //Preferred Check Pattern by SD2.0
   c. If Response is received the card supports High Capacity SD2.0 then send ACMD41 with the following Argument
      Bit[31] = 1'b0; //Reserved bits
      Bit[30] = 1'b1; //High Capacity Status
      Bit[29:24] = 6'h0; //Reserved bits
      Bit[23:0] = Supported Voltage Range
   d. If Response is received for ACMD41 then the card is SD. Otherwise the card is MMC.
   e. If response is not received for initial CMD8 then card does not support High Capacity SD2.0, then issue CMD0 followed by ACMD41 with the following Argument
      Bit[31] = 1'b0; //Reserved bits
      Bit[30] = 1'b0; //High Capacity Status
      Bit[29:24] = 6'h0; //Reserved bits
      Bit[23:0] = Supported Voltage Range
5) Enumerate the card according to the card type.
6) Use a clock source with a frequency = Fod (that is, 400KHz) and use the following enumeration command sequence:

- SD card – Send CMD0, CMD8, ACMD41, CMD2, CMD3.
- MMC – Send CMD0, CMD1, CMD2, CMD3.

3. Power Control

You can implement power control using the following registers, along with external circuitry:

- Control register bits card_voltage_a and card_voltage_b – Status of these bits is reflected at the IO pins. The bits can be used to generate or control the supply voltage that the memory cards require.
- Power enable register – Control power to individual cards.

Programming these two register depends on the implemented external circuitry. While turning on or off the power enable, you should confirm that power supply settings are correct. Power to all cards usually should be disabled while switching off the power.

4. Clock Programming

The Host Controller supports one clock sources. The clock to an individual card can be enabled or disabled. Registers that support this are:

- SDMMC_CLKDIV – Programs individual clock source frequency. SDMMC_CLKDIV limited to 0 or 1 is recommended.
- SDMMC_CLKSRC – Assign clock source for each card.
- SDMMC_CLKENA – Enables or disables clock for individual card and enables low-power mode, which automatically stops the clock to a card when the card is idle for more than 8 clocks.

The Host Controller loads each of these registers only when the start_cmd bit and the Update_clk_regs_only bit in the SDMMC_CMD register are set. When a command is successfully loaded, the Host Controller clears this bit, unless the Host Controller already has another command in the queue, at which point it gives an HLE(Hardware Locked Error). Software should look for the start_cmd and the Update_clk_regs_only bits, and should also set the wait_prvdata_complete bit to ensure that clock parameters do not change during data transfer. Note that even though start_cmd is set for updating clock registers, the Host Controller does not raise a command_done signal upon command completion.

The following shows how to program these registers:

1) Confirm that no card is engaged in any transaction; if there is a transaction, wait until it finishes.
2) Stop all clocks by writing xxxx0000 to the SDMMC_CLKENA register. Set the start_cmd, Update_clk_regs_only, and wait_prvdata_complete bits in the SDMMC_CMD register. Wait until start_cmd is cleared or an HLE is set; in case of an HLE, repeat the command.
3) Program the SDMMC_CLKDIV and SDMMC_CLKSRC registers, as required. Set the start_cmd, Update_clk_regs_only, and wait_prvdata_complete bits in the SDMMC_CMD register. Wait until start_cmd is cleared or an HLE is set; in case of an HLE, repeat the command.
4) Re-enable all clocks by programming the SDMMC_CLKENA register. Set the start_cmd, Update_clk_regs_only, and wait_prvdata_complete bits in the SDMMC_CMD register. Wait until start_cmd is cleared or an HLE is set; in case of an HLE, repeat the command.

5. No-Data Command With or Without Response Sequence

To send any non-data command, the software needs to program the SDMMC_CMD register @0x2C and the SDMMC_CMDARG register @0x28 with appropriate parameters. Using these two registers, the Host Controller forms the command and sends it to the command bus. The Host Controller reflects the errors in the command response through the error bits of the SDMMC_RINTSTS register.

When a response is received – either erroneous or valid – the Host Controller sets the command_done bit in the SDMMC_RINTSTS register. A short response is copied in Response Register0, while along response is copied to all four response registers @0x30, 0x34, 0x38, and 0x3C. The Response3 register bit 31 represents the MSB, and the Response0 register bit 0 represents the LSB of a long response.

For basic commands or non-data commands, follow these steps:

1) Program the Command register @0x28 with the appropriate command argument parameter.
2) Program the Command register @0x2C with the settings in following table.

Table 7-10 Command Settings for No-Data Command

Parameter	Value	Description
Default		
start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used;ref to "use_hold_reg" on SDMMC_CMD register
update_clk_regs_only	0	No clock parameters update command
data_expected	0	No data command
card number	0	Actual card number(one controller only connect one card, the num is No. 0)
cmd_index	command-index	-
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		
wait_prvdata_complete	1	Before sending command on command line, host should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
check_response_crc	1	If host should crosscheck CRC of response received

1) Wait for command acceptance by host. The following happens when the command is loaded into the Host Controller:
   - Host Controller accepts the command for execution and clears the start_cmd bit in the SDMMC_CMD register, unless one command is in process, at which point the Host Controller can load and keep the second command in the buffer.
   - If the Host Controller is unable to load the command – that is, a command is already in progress, a second command is in the buffer, and a third command is attempted – then it generates an HLE (hardware-locked error).
2) Check if there is an HLE.
3) Wait for command execution to complete. After receiving either a response from a card or response timeout, the Host Controller sets the command_done bit in the

SDMMC_RINTSTS register. Software can either poll for this bit or respond to a generated interrupt.

4) Check if response_timeout error, response_CRC error, or response error is set. This can be done either by responding to an interrupt raised by these errors or by polling bits 1, 6, and 8 from the SDMMC_RINTSTS register @0x44. If no response error is received, then the response is valid. If required, the software can copy the response from the response registers @0x30-0x3C.

Software should not modify clock parameters while a command is being executed.

6. Data Transfer Commands

Data transfer commands transfer data between the memory card and the Host Controller. To send a data command, the Host Controller needs a command argument, total data size, and block size. Software can receive or send data through the FIFO.

Before a data transfer command, software should confirm that the card is not busy and is in a transfer state, which can be done using the CMD13 and CMD7 commands, respectively. For the data transfer commands, it is important that the same bus width that is programmed in the card should be set in the card type register @0x18.

The Host Controller generates an interrupt for different conditions during data transfer, which are reflected in the SDMMC_RINTSTS register @0x44 as:

1) Data_Transfer_Over (bit 3) – When data transfer is over or terminated. If there is a response timeout error, then the Host Controller does not attempt any data transfer and the "Data Transfer Over" bit is never set.
2) Transmit_FIFO_Data_request (bit 4) – FIFO threshold for transmitting data was reached; software is expected to write data, if available, in FIFO.
3) Receive_FIFO_Data_request (bit 5) – FIFO threshold for receiving data was reached; software is expected to read data from FIFO.
4) Data starvation by Host timeout (bit 10) – FIFO is empty during transmission or is full during reception. Unless software writes data for empty condition or reads data for full condition, the Host Controller cannot continue with data transfer. The clock to the card has been stopped.
5) Data read timeout error (bit 9) – Card has not sent data within the timeout period.
6) Data CRC error (bit 7) – CRC error occurred during data reception.
7) Start bit error (bit 13) – Start bit was not received during data reception.
8) End bit error (bit 15) – End bit was not received during data reception or for a write operation; a CRC error is indicated by the card.

Conditions 6, 7, and 8 indicate that the received data may have errors. If there was a response timeout, then no data transfer occurred.

7. Single-Block or Multiple-Block Read

Steps involved in a single-block or multiple-block read are:

1) Write the data size in bytes in the SDMMC_BYTCNT register @0x20.
2) Write the block size in bytes in the SDMMC_BLKSIZ register @0x1C. The Host Controller expects data from the card in blocks of size SDMMC_BLKSIZ each.
3) Program the SDMMC_CMDARG register @0x28 with the data address of the beginning of a data read.
4) Program the Command register with the parameters listed in following table. For SD and MMC cards, use CMD17 for a single-block read and CMD18 for a multiple-block read. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

Table 7-11 Command Setting for Single or Multiple-Block Read

Parameter	Value	Description
Default		

Parameter	Value	Description
start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used;ref to "use_hold_reg" on SDMMC_CMD register
update_clk_regs_only	0	No clock parameters update command
card number	0	Actual card number(one controller only connect one card, the num is No.0)
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
send_auto_stop	0/1	-
transfer_mode	0	Block transfer
read_write	0	Read from card
data_expected	1	Data command
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		
cmd_index	command-index	-
wait_prvdata_complete	1	0: Sends command immediately 1: Sends command after previous data transfer ends
check_response_crc	1	0: Host Controller should not check response CRC 1: Host Controller should check response CRC

After writing to the SDMMC_CMD register, the Host Controller starts executing the command; when the command is sent to the bus, the command_done interrupt is generated.

- Software should look for data error interrupts; that is, bits 7, 9, 13, and 15 of the SDMMC_RINTSTS register. If required, software can terminate the data transfer by sending a STOP command.
- Software should look for Receive_FIFO_Data_request and/or data starvation by host timeout conditions. In both cases, the software should read data from the FIFO and make space in the FIFO for receiving more data.
- When a Data_Transfer_Over interrupt is received, the software should read the remaining data from the FIFO.

8. Single-Block or Multiple-Block Write

Steps involved in a single-block or multiple-block write are:

1) Write the data size in bytes in the BYTCNT register @0x20.
2) Write the block size in bytes in the SDMMC_BLKSIZ register @0x1C; the Host Controller sends data in blocks of size SDMMC_BLKSIZ each.

3) Program SDMMC_CMDARG register @0x28 with the data address to which data should be written.
4) Write data in the FIFO; it is usually best to start filling data the full depth of the FIFO.
5) Program the Command register with the parameters listed in following table.

Table 7-12 Command Settings for Single or Multiple-Block Write

Parameter	Value	Description
Default		
start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used;ref to "use_hold_reg" on SDMMC_CMD register
update_clk_regs_only	0	No clock parameters update command
card number	0	Actual card number(one controller only connect one card, the num is No. 0)
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
send_auto_stop	0/1	-
transfer_mode	0	Block transfer
read_write	1	Write to card
data_expected	1	Data command
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		
cmd_index	command-index	-
wait_prvdata_complete	1	0: Sends command immediately 1: Sends command after previous data transfer ends
check_response_crc	1	0: Host Controller should not check response CRC 1: Host Controller should check response CRC

After writing to the SDMMC_CMD register, Host Controller starts executing a command; when the command is sent to the bus, a command_done interrupt is generated.

- Software should look for data error interrupts; that is, for bits 7, 9, and 15 of the SDMMC_RINTSTS register. If required, software can terminate the data transfer by sending the STOP command.
- Software should look for Transmit_FIFO_Data_Request and/or timeout conditions from data starvation by the host. In both cases, the software should write data into the FIFO.
- When a Data_Transfer_Over interrupt is received, the data command is over. For an open-ended block transfer, if the byte count is 0, the software must send the STOP command. If the byte count is not 0, then upon completion of a transfer of a given number of bytes, the Host Controller should send the STOP command, if necessary. Completion of the AUTO-STOP command is reflected by the Auto_command_done interrupt – bit 14 of the SDMMC_RINTSTS register. A response to AUTO_STOP is stored in SDMMC_RESP1 @0x34.

9. Stream Read

A stream read is like the block read mentioned in "Single-Block or Multiple-Block Read", except for the following bits in the Command register:

transfer_mode = 1; //Stream transfer
cmd_index = CMD20;

A stream transfer is allowed for only a single-bit bus width.

10. Stream Write

A stream write is exactly like the block write mentioned in "Single-Block or Multiple-Block Write", except for the following bits in the Command register:

transfer_mode = 1;//Stream transfer
cmd_index = CMD11;

In a stream transfer, if the byte count is 0, then the software must send the STOP command. If the byte count is not 0, then when a given number of bytes completes a transfer, the Host Controller sends the STOP command. Completion of this AUTO_STOP command is reflected by the Auto_command_done interrupt. A response to an AUTO_STOP is stored in the SDMMC_RESP1 register@0x34.
A stream transfer is allowed for only a single-bit bus width.

11. Packed Commands

In order to reduce overhead, read and write commands can be packed in groups of commands—either all read or all write—that transfer the data for all commands in the group in one transfer on the bus.
Packed commands can be of two types:

- Packed Write: CMD23 →CMD25
- Packed Read: CMD23 → CMD25 → CMD23 → CMD18

Packed commands are put in packets by the application software and are transparent to the core.

12. Sending Stop or Abort in Middle of Transfer

The STOP command can terminate a data transfer between a memory card and the Controller, while the ABORT command can terminate an I/O data transfer for only the SDIO_IOONLY and SDIO_COMBO cards.

- Send STOP command – Can be sent on the command line while a data transfer is in progress; this command can be sent at any time during a data transfer.

You can also use an additional setting for this command in order to set the Command register bits (5-0) to CMD12 and set bit 14 (stop_abort_cmd) to 1. If stop_abort_cmd is not set to 1, the Controller does not know that the user stopped a data transfer. Reset bit 13 of the Command register (wait_prvdata_complete) to 0 in order to make the Controller send the command at once, even though there is a data transfer in progress.

- Send ABORT command – Can be used with only an SDIO_IOONLY or SDIO_COMBO card. To abort the function that is transferring data, program the function number in ASx bits (CCCR register of card, address 0x06, bits (0-2) using CMD52.

13. Read_Wait Sequence

Read_wait is used with only the SDIO card and can temporarily stall the data transfer—either from function or memory—and allow the host to send commands to any function within the SDIO device. The host can stall this transfer for as long as required. The Host Controller provides the facility to signal this stall transfer to the card. The steps for doing this are:

1) Check if the card supports the read_wait facility; read SRW (bit 2) of the CCCR register @0x08. If this bit is 1, then all functions in the card support the read_wait facility. Use CMD52 to read this bit.
2) If the card supports the read_wait signal, then assert it by setting the read_wait (bit 6) in the SDMMC_CTRL register @0x00.

3) Clear the read_wait bit in the SDMMC_CTRL register.

14. Controller/DMA/FIFO Reset Usage

- Controller reset – Resets the controller by setting the controller_reset bit (bit 0) in the SDMMC_CTRL register; this resets the CIU and state machines, and also resets the BIU-to-CIU interface. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.
- FIFO reset - Resets the FIFO by setting the fifo_reset bit (bit 1) in the SDMMC_CTRL register; this resets the FIFO pointers and counters of the FIFO. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.

In external DMA transfer mode, even when the FIFO pointers are reset, if there is a DMA transfer in progress, it could push or pop data to or from the FIFO; the DMA itself completes correctly. In order to clear the FIFO, the software should issue an additional FIFO reset and clear any FIFO underrun or overrun errors in the SDMMC_RAWINTS register caused by the DMA transfers after the FIFO was reset.

15. Card Read Threshold

When an application needs to perform a Single or Multiple Block Read command, the application must program the SDMMC_CARDTHRCTL register with the appropriate Card Read Threshold size (CardRdThreshold) and set the Card Read Threshold Enable (CardRdThrEnable) bit to 1'b1. This additional programming ensures that the Host controller sends a Read Command only if there is space equal to the CardRDThreshold available in the Rx FIFO. This in turn ensures that the card clock is not stopped in the middle a block of data being transmitted from the card. The Card Read Threshold can be set to the block size of the transfer, which guarantees that there is a minimum of one block size of space in the RxFIFO before the controller enables the card clock. The Card Read Threshold is required when the Round Trip Delay is greater than 0.5cclk_in period.

16. Error Handling

The Host Controller implements error checking; errors are reflected in the SDMMC_RAWINTS register@0x44 and can be communicated to the software through an interrupt, or the software can poll for these bits. Upon power-on, interrupts are disabled (int_enable in the SDMMC_CTRL register is 0), and all the interrupts are masked (bits 0-31 of the SDMMC_INTMASK register; default is 0).
Error handling:

- Response and data timeout errors – For response timeout, software can retry the command. For data timeout, the Host Controller has not received the data start bit – either for the first block or the intermediate block – within the timeout period, so software can either retry the whole data transfer again or retry from a specified block onwards. By reading the contents of the SDMMC_TCBCNT later, the software can decide how many bytes remain to be copied.
- Response errors – Set when an error is received during response reception. In this case, the response that copied in the response registers is invalid. Software can retry the command.
- Data errors – Set when error in data reception are observed; for example, data CRC, start bit not found, end bit not found, and so on. These errors could be set for any block-first block, intermediate block, or last block. On receipt of an error, the software can issue a STOP or ABORT command and retry the command for either whole data or partial data.
- Hardware locked error – Set when the Host Controller cannot load a command issued by software. When software sets the start_cmd bit in the SDMMC_CMD register, the

Host Controller tries to load the command. If the command buffer is already filled with a command, this error is raised. The software then has to reload the command.

- FIFO underrun/overrun error – If the FIFO is full and software tries to write data in the FIFO, then an overrun error is set. Conversely, if the FIFO is empty and the software tries to read data from the FIFO, an underrun error is set. Before reading or writing data in the FIFO, the software should read the fifo_empty or fifo_full bits in the Status register.
- Data starvation by host timeout – Raised when the Host Controller is waiting for software intervention to transfer the data to or from the FIFO, but the software does not transfer within the stipulated timeout period. Under this condition and when a read transfer is in process, the software should read data from the FIFO and create space for further data reception. When a transmit operation is in process, the software should fill data in the FIFO in order to start transferring data to the card.
- CRC Error on Command – If a CRC error is detected for a command, the CE-ATA device does not send a response, and a response timeout is expected from the Host Controller. The ATA layer is notified that an MMC transport layer error occurred.

Notes: During a multiple-block data transfer, if a negative CRC status is received from the device, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the SDMMC_RINTSTS register. It then continues further data transmission until all the bytes are transmitted.

7.6.4 Voltage Switching

The Host Controller supports SD 3.0 Ultra High Speed (UHS-1) and is capable of voltage switching in SD-mode, which can be applied to SD High-Capacity (SDHC) and SD Extended Capacity (SDXC) cards. UHS-1 supports only 4-bit mode.
However, whether the IO voltage of 1.8v supported or not is depended on the SoC design.
SD 3.0 UHS-1 supports the following transfer speed modes for UHS-50 and/or UHS-104 cards:

- DS – default-speed up to 25MHz, 3.3V signaling
- HS – high-speed up to 50MHz, 3.3V signaling
- SDR12 – SDR up to SDR 25MHz, 1.8V signaling
- SDR25 – SDR up to 50MHz, 1.8V signaling
- SDR50 – SDR up to 100MHz, 1.8V signaling
- DDR50 – DDR up to 50MHz, 1.8V signaling

Voltage selection can be done in only SD mode. The first CMD0 selects the bus mode-either SD mode or SPI mode. The card must be in SD mode in order for 1.8V signaling mode to apply, during which time the card cannot be switched to SPI mode or 3.3V signaling without a power cycle.
If the System BIOS in an embedded system already knows that it is connected to an SD 3.0 card, then the driver programs the Controller to initiate ACMD41. The software knows from the response of ACMD41 whether or not the card supports voltage switching to 1.8V.

- If bit 32 of ACMD41 response is 1'b1: card supports voltage switching and next command-CMD11-invokes voltage switching sequence. After CMD11 is started, the software must program the IO voltage selection register based on the soc architecture.
- If bit 32 of ACMD41 response is 1'b0: card does not support voltage switching and CMD11 should not be started.

If the card and host controller accept voltage switching, then they support UHS-1 modes of data transfer. After the voltage switch to 1.8V, SDR12 is the default speed.
Since the UHS-1 can be used in only 4-bit mode, the software must start ACMD6 and change the card data width to 4-bit mode; ACMD6 is driven in any of the UHS-1 speeds. If the host wants to select the DDR mode of data transfer, then the software must program the SDMMC_DDR_REG register in the CSR space with the appropriate card number.
To choose from any of the SDR or DDR modes, appropriate values should be programmed

in the SDMMC_CLKDIV register.

1. Voltage Switch Operation

The Voltage Switch operation must be performed in SD mode only.

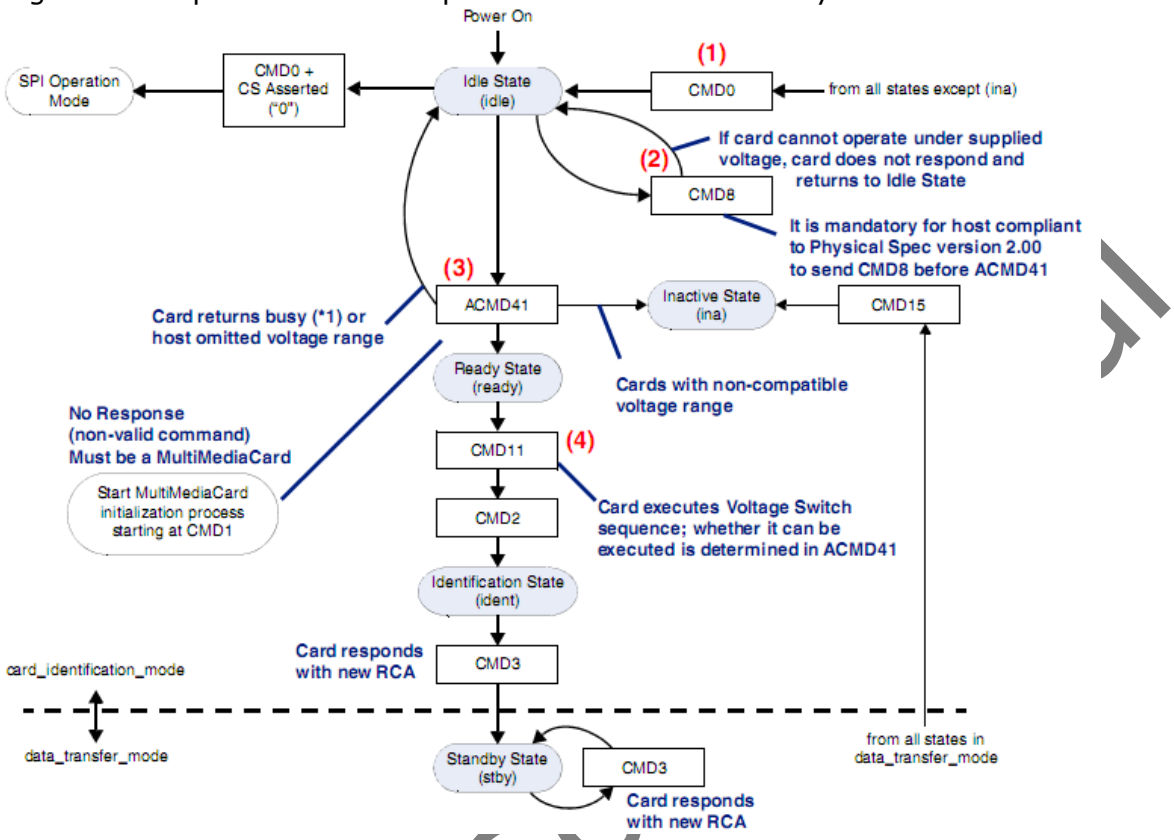

Fig. 7-11 Voltage Switching Command Flow Diagram

The following outlines the steps for the voltage switch programming sequence

1) Software Driver starts CMD0, which selects the bus mode as SD.
2) After the bus is in SD card mode, CMD8 is started in order to verify if the card is compatible with the SD Memory Card Specification, Version 2. 00. CMD8 determines if the card is capable of working within the host supply voltage specified in the VHS (19:16) field of the CMD; the card supports the current host voltage if a response to CMD8 is received.
3) ACMD 41 is started. The response to this command informs the software if the card supports voltage switching; bits 38, 36, and 32 are checked by the card argument of ACMD41; refer to following figure.

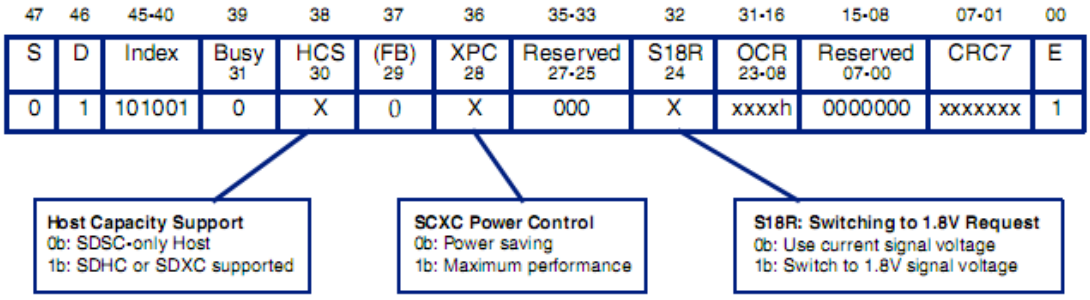

Fig. 7-12 ACMD41 Argument

- Bit 30 informs the card if host supports SDHC/SDXC or not; this bit should be set to 1'b1.
- Bit 28 can be either 1 or 0.
- Bit 24 should be set to 1'b1, indicating that the host is capable of voltage switching; refer to following figure.

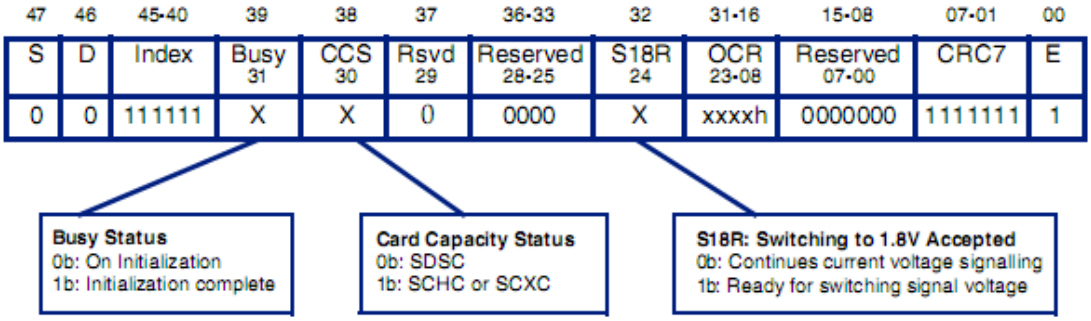

Fig. 7-13 ACMD41 Response(R3)

- Bit 30 – If set to 1'b1, card supports SDHC/SDXC; if set to 1'b0, card supports only SDSC
- Bit 24 – If set to 1'b1, card supports voltage switching and is ready for the switch
- Bit 31 – If set to 1'b1, initialization is over; if set to 1'b0, means initialization in process

4) If the card supports voltage switching, then the software must perform the steps discussed for either the "Voltage Switch Normal Scenario" or the "Voltage Switch Error Scenario".

2. Voltage Switch Normal Scenario

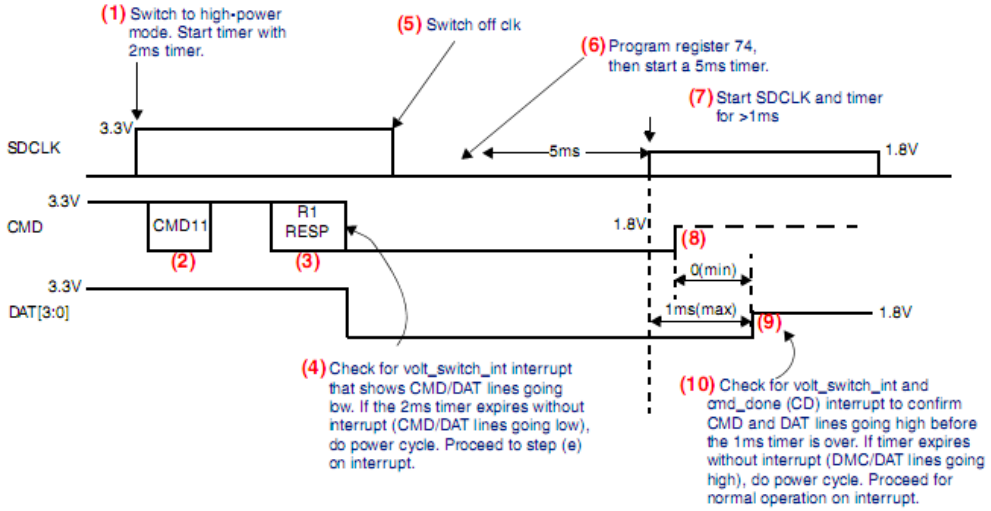

Fig. 7-14 Voltage Switch Normal Scenario

1) The host programs SDMMC_CLKENA—cclk_low_power register—with zero (0) for the corresponding card, which makes the host controller move to high-power mode. The application should start a timer with a recommended value of 2ms; this value of 2 ms is determined as below:Total clk required for CMD11 = 48 clks
Total clk required for RESP R1 = 48 clks
Maximum clk delay between MCD11 end to start of RESP1 = 60 clks
Total = 48+48 + 60 = 160
Minimum frequency during enumeration is 100 KHz; that is, 10us
Total time = 160 * 10us = 1600us = 1. 6ms ~ 2ms
2) The host issues CMD11 to start the voltage switch sequence. Set bit 28 to 1'b1 in CMD when setting CMD11; for more information on setting bits, refer to "Boot Operation".
3) The card returns R1 response; the host controller does not generate cmd_done interrupt on receiving R1 response.
4) The card drives CMD and DAT [3:0] to low immediately after the response. The host controller generates interrupt (VOLT_SWITCH_INT) once the CMD or DAT [3:0] line goes low. The application should wait for this interrupt. If the 2ms timer expires without an interrupt (CMD/DAT lines going low), do a power cycle.

Note: Before doing a power cycle, switch off the card clock by programming SDMMC_CLKENA register
Proceed to step (5) on getting an interrupt (VOLT_SWITCH_INT).
Note: This interrupt must be cleared once this interrupt is received. Additionally, this interrupt should not be masked during the voltage switch sequence.
If the timer expires without interrupt (CMD/DAT lines going low), perform a power cycle. Proceed to step (5) on interrupt.

1) Program the SDMMC_CLKENA, cclk_enable register, with 0 for the corresponding card; the host stops supplying SDCLK.
2) Program Voltage register to the required values for the corresponding card. The application should start a timer > 5ms.
3) After the 5ms timer expires, the host voltage regulator is stable. Program SDMMC_CLKENA, cclk_enable register, with 1 for the corresponding card; the host starts providing SDCLK at 1. 8V; this can be at zero time after Voltage register has been programmed. When the SDMMC_CLKENA register is programmed, the application should start another timer > 1ms.
4) By detecting SDCLK, the card drives CMD to high at 1. 8V for at least one clock and then stops driving (tri-state); CMD is triggered by the rising edge of SDCLK (SDR timing).
5) If switching to 1. 8V signaling is completed successfully, the card drives DAT [3:0] to high at 1. 8V for at least one clock and then stops driving (tri-state); DAT [3:0] is triggered by the rising edge of SDCLK (SDR timing). DAT[3:0] must be high within 1ms from the start of SDCLK.
6) The host controller generates a voltage switch interrupt (VOLT_SWITCH_INT) and a command done (CD) interrupt once the CMD and DAT[3:0] lines go high. The application should wait for this interrupt to confirm CMD and DAT lines going high before the 1ms timer is done.

If the timer expires without the voltage switch interrupt (VOLT_SWITCH_INT), a power cycle should be performed. Program the SDMMC_CLKENA register to stop the clock for the corresponding card number. Wait for the cmd_done (CD) interrupt. Proceed for normal operation on interrupt. After the sequence is completed, the host and the card start communication in SDR12 timing.

3. Voltage Switch Error Scenario

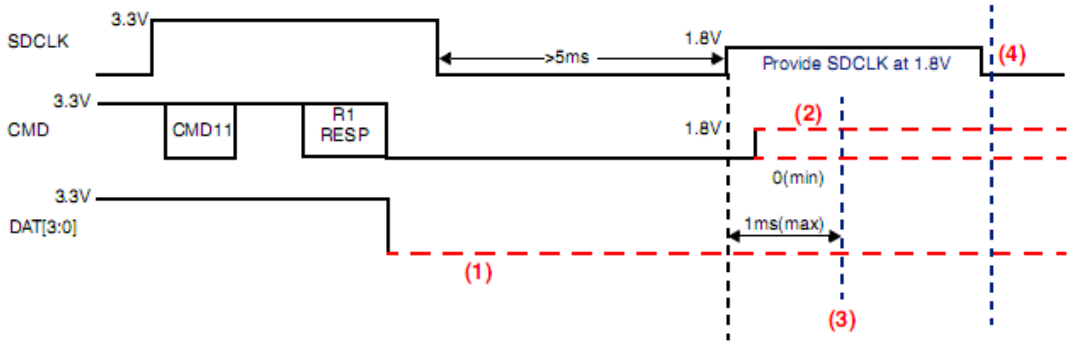

Fig. 7-15 Voltage Switch Error Scenario

1) If the interrupt (VOLT_SWITCH_INT) does not come, then the 2 ms timer should time out and a power cycle should be initiated.

Note: Before performing a power cycle, switch off the card clock by programming SDMMC_CLKENA register; no cmd_done (CD) interrupt is generated.
Additionally, if the card detects a voltage error at any point in between steps (5) and (7) in the card keeps driving DAT[3:0] to low until card power off.

2) CMD can be low or tri-state.

3) The host controller generates a voltage switch interrupt once the CMD and DAT[3:0] lines go high. The application should check for an interrupt to confirm CMD and DAT lines going high before the 1 ms timer is done.

If the 1 ms timer expires without interrupt (VOLT_SWITCH_INT) and cmd_done (CD), a power cycle should be performed. Program the SDMMC_CLKENA register to stop SDCLK of the corresponding card. Wait for the cmd_done interrupt. Proceed for normal operation on interrupt.

4) If DAT[3:0] is low, the host drives SDCLK to low and then stops supplying the card power.

Note: The card checks voltages of its own regulator output and host signals to ensure they are less than 2.5V. Errors are indicated by (1) and (2).

- If voltage switching is accepted by the card, the default speed is SDR12.
- Command Done is given:
  - If voltage switching is properly done, CMD and DAT line goes high.
  - If switching is not complete, the 1ms timer expires, and the card clk is switched off.

Note: No other CMD should be driven before the voltage switching operation is completed and Command Done is received.

- The application should use CMD6 to check and select the particular function; the function appropriate-speed should be selected.

After the function switches, the application should program the correct value in the CLKDIV register, depending on the function chosen. Additionally, if Function 0x4 of the Access mode is chosen—that is, DDR50, then the application should also program 1'b1 in DDR_REG for the card number that has been selected for DDR50 mode.

7.6.5 Back-End Power

Each device needs one bit to control the back-end power supply for an embedded device; this bit does not control the VDDH of the host controller. A back_end_power register enables software programming for back-end power. The value on this register is output to the back_end_power signal, which can be used to switch power on and off the embedded device.

7.6.6 H/W Reset Operation

When the RST_n signal goes low, the card enters a pre-idle state from any state other than the inactive state.

H/W Reset Programming Sequence

The following outlines the steps for the H/W reset programming sequence:

11) Program CMD12 to end any transfer in process.
12) Wait for DTO, even if no response is sent back by the card.
13) Set the following resets:

- DMA reset–SDMMC_CTRL[2]
- FIFO reset –SDMMC_CTRL[1] bits

Note: The above steps are required only if a transfer is in process.

14) Program the CARD_RESET register with a value of 0; this can be done at any time when the card is connected to the controller. This programming asserts the RST_n signal and resets the card.
15) Wait for minimum of 1 µs or cclk_in period, whichever is greater
16) After a minimum of 1 µs, the application should program a value of 0 into the CARD_RESET register. This de-asserts the RST_n signal and takes the card out of reset.
17) The application can program a new CMD only after a minimum of 200 µs after the de-assertion of the RST_n signal, as per the MMC 4.41 standard.

Note: For backward compatibility, the RST_n signal is temporarily disabled in the card by default. The host may need to set the signal as either permanently enabled or permanently disabled before it uses the card.

7.6.7 FBE Scenarios

An FBE occurs due to an AHB error response on the AHB bus. This is a system error, so the software driver should not perform any further programming to the Host. The only recovery mechanism from such scenarios is to do one of the following:

- Issue a hard reset by asserting the reset_n signal
- Do a program controller reset by writing to the CTRL[0] register

1. FIFO Overflow and Underflow

During normal data transfer conditions, FIFO overflow and underflow will not occur. However if there is a programming error, then FIFO overflow/underflow can result. For example, consider the following scenarios.

- For transmit: PBL=4, Tx watermark = 1. For the above programming values, if the FIFO has only one location empty, it issues a dma_req to IDMAC FSM. Due to PBL value=4, the IDMAC FSM performs 4 pushes into the FIFO. This will result in a FIFO overflow interrupt.
- For receive: PBL=4, Rx watermark = 1. For the above programming values, if the FIFO has only one location filled, it issues a dma_req to IDMAC FSM. Due to PBL value=4, the IDMAC FSM performs 4 pops to the FIFO. This will result in a FIFO underflow interrupt.

The driver should ensure that the number of bytes to be transferred as indicated in the descriptor should be a multiple of 4bytes with respect to H_DATA_WIDTH=32. For example, if the BYTCNT = 13, the number of bytes indicated in the descriptor should be 16 for H_DATA_WIDTH=32.

2. Programming of PBL and Watermark Levels

The DMAC performs data transfers depending on the programmed PBL and threshold values.

Table 7-13 PBL and Watermark Levels

PBL (Number of transfers)	Tx/Rx Watermark Value
1	greater than or equal to 1
4	greater than or equal to 4
8	greater than or equal to 8
16	greater than or equal to 16
32	greater than or equal to 32
64	greater than or equal to 64
128	greater than or equal to 128
256	greater than or equal to 256

7.6.8 Variable Delay/Clock Generation

Variable delay mechanism for the cclk_in_drv is optional, but it can be useful in order to meet a range of hold-time requirements across modes. Variable delay mechanism for the cclk_in_sample is mandatory and is required to achieve the correct sampling point for data. cclk_in/cclk_in_sample/ cclk_in_drv is generated by Clock Generation Unit (CLKGEN) with variable delay mechanism, which includes Phase Shift Unit and Delay Line Unit selectable. The Phase Shift Unit can shift cclk_in_sample/cclk_in_drv by 0/90/180/270-degree relative to cclk_in, controlled by sample_degree/drv_degree.

The Delay Line Unit can shift cclk_in_sample/cclk_in_drv in the unit of 40ps~80ps for every delay element. The delay unit number is determined by sample_delaynum/drv_delaynum, and enabled by sample_sel/drv_sel.

cclk_in is generated by cclkin divided by 2. cclk_in_drv and cclk_in_sample clocks are phase-shifted with delayed versions of cclk_in. All clocks are recommended to have a 50% duty cycle; DDR modes must have 50% duty cycles.

The architecture is as follows.

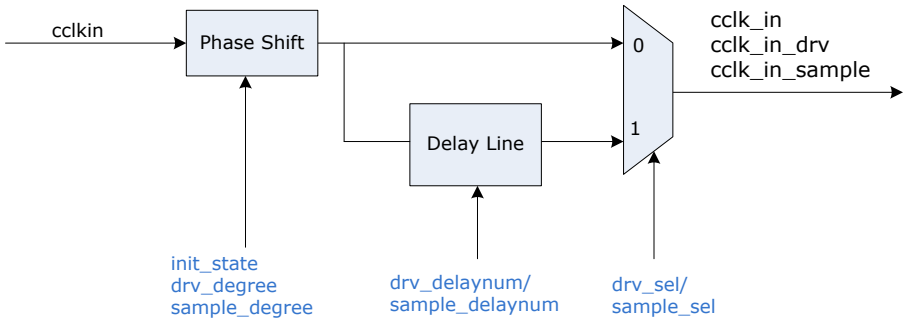

Fig. 7-16 Clock Generation Unit

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

Table 7-14 Configuration for SDMMC Clock Generation

Signal Name	Source	Default	Description
init_state	CRU_SDMMC_CON00[0]	0	Soft initial state for phase shift.
drv_degree[1:0]	CRU_SDMMC_CON00[2:1]	2	Phase shift for cclk_in_drv. 0: 0-degree 1: 90-degree 2: 180-degree 3: 270-degree
drv_delaynum[7:0]	CRU_SDMMC_CON00[10:3]	0	Element number in delay line for cclk_in_drv
drv_sel	CRU_SDMMC_CON00[11]	0	cclk_in_drv source selection: 0: use clock after phase_shift 1: use clock after phase_shift and delay line
sample_degree[1:0]	CRU_SDMMC_CON01[2:1]	0	Phase shift for cclk_in_sample. 0: 0-degree 1: 90-degree 2: 180-degree 3: 270-degree
sample_delaynum[7:0]	CRU_SDMMC_CON01[10:3]	0	Element number in delay line for cclk_in_sample
sample_sel	CRU_SDMMC_CON01[11]	0	cclk_in_sample source selection: 0: use clock after phase_shift 1: use clock after phase_shift and delay line

The following outlines the steps for clock generation sequence:

1) Assert init_state to soft reset the CLKGEN.
2) Configure drv_degree/sample_degree.
3) If fine adjustment required, delay line can be used by configuring drv_delaynum/sample_delaynum and drv_sel/sample_sel.
4) Dis-assert init_state to start CLKGEN.

7.6.9 Variable Delay Tuning

Tuning is defined by SD and MMC cards to determine the correct sampling point required for the host, especially for the speed modes SDR104 and HS200 where the output delays from the cards can be up to 2 UI. Tuning is required for other speed modes-such as DDR50-even though the output delay from the card is less than one cycle.
Command for tuning is different for different cards.

- SD Memory Card:
  - CMD19 – SD card for SDR50 and SDR104 speed modes. Tuning data is defined by card specifications.
  - CMD6 – SD card for speed modes not supporting CMD19. Tuning data is the 64byte SD status.

- Multimedia Card:
  - CMD21 – MMC card for HS200 speed mode. Tuning data is defined by card specifications.
  - CMD8 – MMC card for speed modes not supporting CMD21. Tuning data is 512 byte ExtCSD data.

The following is the procedure for variable delay tuning:

1) Set a phase shift of 0-degree on cclk_in_sample.
2) Send the Tuning command to the card; the card in turn sends an R1 response on the CMD line and tuning data on the DAT line.
3) If the host sees any of the errors—start bit error, data crc error, end bit error, data read time-out, response crc error, response error—then the sampling point is incorrect.
4) Send CMD12 to bring the host controller state machines to idle.

- The card may treat CMD12 as an invalid command because the card has successfully sent the tuning data, and it cannot send a response.
- The host controller may generate a response time-out interrupt that must be cleared by software.

5) Repeat steps 2) to 4) by increasing the phase shift value or delay element number on cclk_in_sample until the correct sampling point is received such that the host does not see any of the errors.
6) Mark this phase shift value as the starting point of the sampling window.
7) Repeat steps 2 to 4 by increasing the phase shift value or delay element number on cclk_in_sample until the host sees the errors starting to come again or the phase shift value reaches 360-degree.
8) Mark the last successful phase shift value as the ending point of the sampling window.

A window is established where the tuning block is matched. For example, for a scenario where the tuning block is received correctly for a phase shift window of 90-degree and 180-degree, then an appropriate sampling point is established as 135-degree. Once a sampling point is established, no errors should be visible in the tuning block.

7.6.10 Package Command

In order to reduce overhead, read and write commands can be packed in groups of commands-either all read or all write-that transfer the data for all commands in the group in one transfer on the bus.

Packed commands can be of two types:

- Packed Write: CMD23 → CMD25
- Packed Read: CMD23 → CMD25 → CMD23 → CMD18

Packed commands are put in packets by the application software and are transparent to the core. For more information on packed commands, refer to the eMMC specification.

7.6.11 Card Detection Method

As the interface description shows, SD/MMC does not have detect IO, so it is necessary to use GPIO as detect IO for SDMMC device detection.

Chapter 8 USB2.0 OTG

8.1 Overview

USB2.0 OTG is a Dual-Role Device controller, which supports both device and host functions and is fully compliant with OTG Supplement to USB2.0 specification and support high-speed (480Mbps), full-speed (12Mbps), low-speed (1.5Mbps) transfer.
USB2.0 OTG is optimized for portable electronic devices, point-to-point applications (no hub, direct connection to device) and multi-point applications to devices.

8.1.1 Features

- Compliant with the OTG Supplement to the USB2.0 Specification
- Operates in High-Speed, Full-Speed and Low-Speed mode (host mode only)
- Support 9 channels in host mode
- 9 Device mode endpoints in addition to control endpoint 0, 4 in, 3 out and 2 IN/OUT
- Built-in one 1024x35 bits FIFO
- Internal DMA with scatter/gather function
- Supports packet-based, dynamic FIFO memory allocation for endpoints for flexible, efficient use of RAM
- Support dynamic FIFO sizing

8.2 Block Diagram

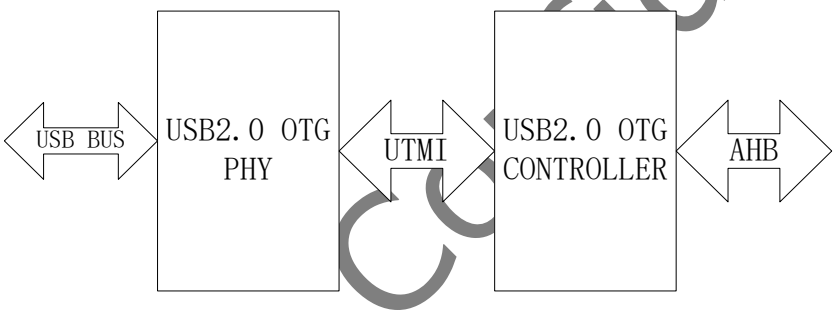

Fig.8-1 USB2.0 OTG Architecture

The Fig shows the architecture of USB2.0 OTG. It is broken up into two separate units: USB2.0 OTG controller and USB2.0 OTG PHY. The two units are interconnected with UTMI interface.

8.3 USB2.0 OTG Controller

The USB2.0 OTG Controller can active as USB2.0 host controller or USB2.0 device controller basing on ID status of Micron-AB receptacle.
As USB2.0 host controller, it uses one transmit FIFO for all non-periodic out transactions and one transmit FIFO for all periodic out transactions as transmit buffers to hold the data to be transmitted over USB, and use one receive FIFO for all periodic and non-periodic transactions to hold the received data from the USB until it is transferred to the system memory by DMA.
As USB2.0 device controller, it also use a single receive FIFO to receive the data for all the out endpoints from USB, and individual transmit FIFOs for each IN endpoint.

8.4 USB2.0 OTG PHY

The USB2.0 OTG PHY handles the low level USB protocol and signaling from controller and differential pairs. It includes functions such as data serialization and de-serialization, bit stuffing and clock recovery and synchronization. Its feature contains:

- provide dual UTMI ports
- OTG0 Support UART Bypass Function
- Fully compliant with USB specifications Rev 2.0
- Supports 480Mbps (HS), 12Mbps (FS) & 1.5Mbps (LS) serial data transmission

- Supports low latency hub mode with 40 bit time around trip delay
- 16 bit UTMI interface compliant with UTMI+ specification level 3 Rev 1.
- Loop back BIST mode supported
- Built-in I/O and ESD structure
- On-die self-calibrated HS/FS/LS termination

8.5 Register Description

Please refer to the document DWC_otg_databook.pdf

8.6 Interface description

Table 8-1USB2.0 OTG Interface Description

Module Pin	Direction	Pad Name	pinmux
DM	A	IO_OTG_DM	-
DP	A	IO_OTG_DP	-
VBUS	A	IO_OTG_VBUS	-
ID	A	IO_OTG_ID	-

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

8.7 Application Note

8.7.1 Suspend Mode

When USB2.0 OTG PHY is in suspend state

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

Please refer to "Chapter GRF" for configuration details.

8.7.2 Relative GRF Registers

USB2.0 OTG PHY contains some registers to configure. These bits are used to adjust DP/DM SI. Please refer to "Chapter GRF" for more details.

Chapter 9 Audio PWM

9.1 Overview

The Audio PWM provides an easy and cheap solution for audio playback in low quality. It acts as a digital-to-analog converter (DAC), which converts the digital audio PCM data to the analog PWM signals.

The Audio PWM supports the following features:

- Support one 32bit AHB slave interface for the configuration and data access.
- Support one 32x32bits TX FIFO
- The FIFO can be written by Cortex-M4, Cortex-M0, HiFi3 and DMAC
- Support 16bits~32bits source data width
- Support 8bits~11bits output sampling data width
- Support two maskable output channels, left and right, and they can be swapped
- Support the interpolation rate up to 15
- Support linear interpolation, and the interpolation rate must be 1/3/7/15 in this mode
- Support one interrupt output for FIFO almost full, overrun and empty

9.2 Block Diagram

The following figure shows the block diagram of Audio PWM.

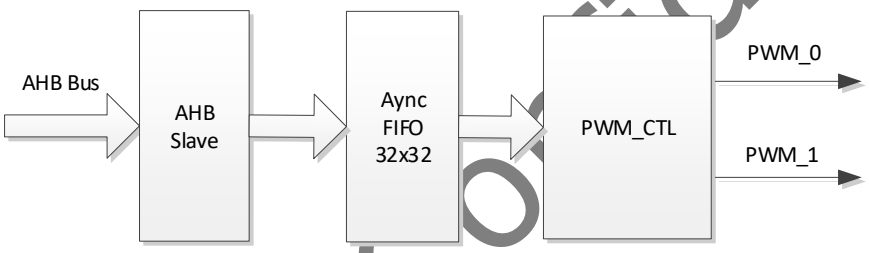

Fig. 9-1 Block Diagram of Audio PWM

9.3 Function Description

When the source data width is 16 bits, all the 32 bits of the FIFO data is valid in normal cases. The lower 16 bits are used for the channel 0 and the upper 16 bits are used for the channel 1. And when the source data width is greater than 16 bits, the word-wide data is used alternately for the channel 0 and the channel 1.

The Audio PWM samples two halfwords/words of the FIFO data for the two output channels, and the interpolation calculation also needs the next two halfwords/words. Therefore, the four sampling data values which stored in the four data buffers are needed at the beginning of the transfer. And it is required that the next two sampling data values are prefetched from the FIFO later. For this reason, when the transfer is finish by the long stop operation and the FIFO is empty, there are still two or four unprocessed words left in the buffers.

The Pulse Width Modulation (PWM) is a form of signal modulation where data is represented by the ratio of the high level time to the period, which is known as the duty cycle. In this chip, the PWM period is defined as (2^N-1) clock cycles, where N is the output sampling data width of the Audio PWM, which determines the duty cycle resolution and the required frequency of the system main clock. The duty cycle of 0% represents the minimum signed digit while the maximum signed digit is represented by the duty cycle of 100%.

The following table shows the duty cycle resolution of the Audio PWM.

Table 9-1 Duty Cycle Resolution of the Audio PWM

Output Width	Duty Cycle Resolution	Duty Cycle Range	Represented Signed Digit
8	256	0/255 ~ 255/255	-128 ~ 127
9	512	0/511 ~ 511/511	-256 ~ 255
10	1024	0/1023 ~ 1023/1023	-512 ~ 511
11	2048	0/2047 ~ 2047/2047	-1024 ~ 1023

9.4 Register Description

This section describes the control/status registers of the design. Software should read and write these registers using 32-bits accesses.

9.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
AUDPWM_VERSION	0x0000	W	0x01000000	Version Number Register
AUDPWM_XFER	0x0004	W	0x00000000	Transfer Control Register
AUDPWM_SRC_CFG	0x0008	W	0x00000000	Source Data Configuration Register
AUDPWM_PWM_CFG	0x0010	W	0x00000000	PWM Configuration Register
AUDPWM_PWM_ST	0x0014	W	0x00000000	PWM Status Register
AUDPWM_PWM_BUF_01	0x0018	W	0x00000000	PWM Processed Data Buffer 0 & 1 Register
AUDPWM_PWM_BUF_23	0x001c	W	0x00000000	PWM Processed Data Buffer 2 & 3 Register
AUDPWM_FIFO_CFG	0x0020	W	0x00000000	FIFO Configuration Register
AUDPWM_FIFO_LVL	0x0024	W	0x00000000	FIFO Level Register
AUDPWM_FIFO_INT_EN	0x0028	W	0x00000000	FIFO Interrupt Enable Register
AUDPWM_FIFO_INT_ST	0x002c	W	0x00000000	FIFO Interrupt Status Register
AUDPWM_FIFO_ENTRY	0x0030	W	0x00000000	FIFO Data Entry Register

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

9.4.2 Detail Register Description

AUDPWM_VERSION

Address: Operational Base + offset (0x0000)

Bit	Attr	Reset Value	Description
31:0	RO	0x01000000	version Version number: 1.0.0.0

AUDPWM_XFER

Address: Operational Base + offset (0x0004)

Bit	Attr	Reset Value	Description
31:18	RO	0x0	reserved
17:16	WO	0x0	write_mask Write enable for lower 16 bits, each bit is individual. 1'b0: Write access disable 1'b1: Write access enable

Bit	Attr	Reset Value	Description
15:2	RO	0x0	reserved
1	RW	0x0	lstop Long stop. Stop when read out all FIFO data, but there are still 2 or 4 unsent words left in the buffers. 1'b0: Nothing 1'b1: Long stop
0	RW	0x0	start Transfer start or stop. 1'b0: Stop transfer 1'b1: Start transfer

AUDPWM_SRC_CFG

Address: Operational Base + offset (0x0008)

Bit	Attr	Reset Value	Description
31:23	RO	0x0	reserved
22:16	WO	0x00	write_mask Write enable for lower 16 bits, each bit is individual. 1'b0: Write access disable 1'b1: Write access enable
15:7	RO	0x0	reserved
6	RW	0x0	half_en Half mode enable. This bit field is significant only when 16 bits source data width is selected. 1'b0: All the 32 bits of data is valid (The lower 16 bits for channel 0 and the upper 16 bits for channel 1) 1'b1: Only the lower 16 bits of data is valid and the upper 16 bits of data is invalid
5	RW	0x0	align Source data alignment. This bit field is significant when 16 bits source data width is not selected. 1'b0: Right aligned 1'b1: Left aligned

Bit	Attr	Reset Value	Description
4:0	RW	0x00	width Source data width. 5'd0~5'd14: Reserved 5'd15: 16 bits 5'd16: 17 bits 5'd17: 18 bits 5'd18: 19 bits 5'd19: 20 bits 5'd20: 21 bits 5'd21: 22 bits 5'd22: 23 bits 5'd23: 24 bits 5'd24: 25 bits 5'd25: 26 bits 5'd26: 27 bits 5'd27: 28 bits 5'd28: 29 bits 5'd29: 30 bits 5'd30: 31 bits 5'd31: 32 bits

AUDPWM_PWM_CFG

Address: Operational Base + offset (0x0010)

Bit	Attr	Reset Value	Description
31:26	RO	0x0	reserved
25:16	WO	0x000	write_mask Write enable for lower 16 bits, each bit is individual. 1'b0: Write access disable 1'b1: Write access enable
15:10	RO	0x0	reserved
9:8	RW	0x0	sample_width Output sampling data width. 2'b00: 8 bits 2'b01: 9 bits 2'b10: 10 bits 2'b11: 11 bits
7	RW	0x0	right_dis Right channel output disable. 1'b0: Enable 1'b1: Disable
6	RW	0x0	left_dis Left channel output disable. 1'b0: Enable 1'b1: Disable

Bit	Attr	Reset Value	Description
5	RW	0x0	out_swap Output swap. 1'b0: Audio PWM channel 0 for left channel and Audio PWM channel 1 for right channel (Default output IO map) 1'b1: Audio PWM channel 1 for left channel and Audio PWM channel 0 for right channel (Swap output IO map)
4	RW	0x0	linear_interp_en Linear interpolation mode enable. Note that linear interpolation only support interpolation rate = 1/3/7/15. 1'b0: Disable 1'b1: Enable
3:0	RW	0x0	interp_rate Interpolation rate. 4'd0: Interpolate 0 point 4'd1: Interpolate 1 point 4'd2: Interpolate 2 points 4'd3: Interpolate 3 points 4'd4: Interpolate 4 points 4'd5: Interpolate 5 points 4'd6: Interpolate 6 points 4'd7: Interpolate 7 points 4'd8: Interpolate 8 points 4'd9: Interpolate 9 points 4'd10: Interpolate 10 points 4'd11: Interpolate 11 points 4'd12: Interpolate 12 points 4'd13: Interpolate 13 points 4'd14: Interpolate 14 points 4'd15: Interpolate 15 points

AUDPWM_PWM_ST

Address: Operational Base + offset (0x0014)

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RO	0x0	pwm_busy Audio PWM status. 1'b0: Audio PWM is idle 1'b1: Audio PWM is busy
0	RO	0x0	fifo_busy TX FIFO status. 1'b0: FIFO is idle 1'b1: FIFO is busy

AUDPWM_PWM_BUF_01

Address: Operational Base + offset (0x0018)

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:16	RO	0x000	pwm_buf_1 Current data value of channel 1.
15:11	RO	0x0	reserved
10:0	RO	0x000	pwm_buf_0 Current data value of channel 0.

AUDPWM_PWM_BUF_23

Address: Operational Base + offset (0x001c)

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:16	RO	0x000	pwm_buf_3 Next data value of channel 1.
15:11	RO	0x0	reserved
10:0	RO	0x000	pwm_buf_2 Next data value of channel 0.

AUDPWM_FIFO_CFG

Address: Operational Base + offset (0x0020)

Bit	Attr	Reset Value	Description
31:28	RO	0x0	reserved
27:16	WO	0x000	write_mask Write enable for lower 16 bits, each bit is individual. 1'b0: Write access disable 1'b1: Write access enable
15:13	RO	0x0	reserved
12:8	RW	0x00	almost_full_watermark TX FIFO almost full watermark. When the number of valid FIFO data entries is less than or equal to the value in this bit field, the almost full interrupt is triggered.
7	RW	0x0	dma_en DMA TX request enable. 1'b0: Disable 1'b1: Enable
6:5	RO	0x0	reserved
4:0	RW	0x00	dma_watermark DMA watermark. This bit field controls the level at which a DMA request is made by the transmit logic. The watermark level = watermark + 1; that is, dma_tx_req is generated when the number of valid FIFO data entries is greater than the value in this field.

AUDPWM_FIFO_LVL

Address: Operational Base + offset (0x0024)

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5:0	RO	0x00	fifo_space2full TX FIFO space2full level. Indicates the number of valid data entries in the TX FIFO.

AUDPWM_FIFO_INT_EN

Address: Operational Base + offset (0x0028)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	empty_int_en TX FIFO empty interrupt enable. 1'b0: Disable 1'b1: Enable
1	RW	0x0	overrun_int_en TX FIFO overrun interrupt enable. 1'b0: Disable 1'b1: Enable
0	RW	0x0	full_int_en TX FIFO almost full interrupt enable. 1'b0: Disable 1'b1: Enable

AUDPWM_FIFO_INT_ST

Address: Operational Base + offset (0x002c)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	empty_int_st TX FIFO empty interrupt status. 1'b0: Nothing 1'b1: FIFO empty
1	RW	0x0	overrun_int_st TX FIFO overrun interrupt status. 1'b0: Nothing 1'b1: FIFO overrun
0	RO	0x0	full_int_st TX FIFO almost full interrupt status. 1'b0: Nothing 1'b1: FIFO almost full

AUDPWM_FIFO_ENTRY

Address: Operational Base + offset (0x0030)

Bit	Attr	Reset Value	Description
31:0	WO	0x00000000	fifo_data_entry External DMAC or application writes data to this address.

9.5 Interface Description

Table 9-2 Audio PWM Interface Description

Module Pin	Direction	Pin Name	IOMUX Setting
AUDPWM_L	O	PWM0_M1/UART0_CTSN_M0/SPI0_CS0n_M0/I2C0_SDA_M0/TKEY0_M0/TKEY_DRIVE_M0/PWM_AUDIO_L_M0/I2C2_SDA_M1/GPIO0_B4_d	GRF_GPIO0B_IOMUX_H[3:0]=4'b0111
		SRADC6/PWM6_M0/UART0_RX_M1/SPI1_MOSI_M0/I2C0_SDA_M1/I2S_SDO_M0/TKEY7/PWM_AUDIO_L_M1/GPIO0_C6_u	GRF_GPIO0C_IOMUX_H[11:8]=4'b1000
AUDPWM_R	O	PWM1_M1/UART0_RTSN_M0/SPI0_CLK_M0/I2C0_SCL_M0/TKEY1_M0/PWM_AUDIO_R_M0/I2C2_SCL_M1/GPIO0_B5_d	GRF_GPIO0B_IOMUX_H[6:4]=3'b110
		SRADC7/PWM7_M0/UART0_TX_M1/SPI1_MISO_M0/I2C0_SCL_M1/I2S_SDI_M0/TKEY8/PWM_AUDIO_R_M1/PMIC_INT_M1/GPIO0_C7_d	GRF_GPIO0C_IOMUX_H[15:12]=4'b1000

Notes: Unused Module Pin is tied to zero! I=input, O=output, I/O=input/output, bidirectional

9.6 Application Notes

- When the AUDPWM_FIFO_CFG.dma_en is set to 1 and the number of the valid FIFO data entries is greater than AUDPWM_FIFO_CFG.dma_watermark, the Audio PWM will generate a DMA request to the external DMAC. And then the required word-aligned data will be written to the FIFO if the DMAC has been correctly configured.
- Start the data transfer and output the processed PWM signals to the IO by configuring AUDPWM_XFER.start to 1. If this bit is set to 0, the processing will terminate immediately.
- Generally, you can also stop the processing when all the FIFO data has been read out by configuring AUDPWM_XFER.lstop to 1. In this way, the long stop bit will be auto cleared to 0 when the FIFO is empty. You should wait for the FIFO and the control logic of the Audio PWM to be idle by polling AUDPWM_PWM_ST.
- If the FIFO is empty but the processing is still running, the output level will always be low.
- You can enable the FIFO interrupt by configuring AUDPWM_FIFO_INT_EN and query the interrupt status from AUDPWM_FIFO_INT_ST. Writing 1 to the interrupt status bit of overrun or empty will clear it to 0, but the bit of almost full cannot be cleared by this means. Note that the FIFO overrun interrupt cannot occur in application.
- When the linear interpolation mode is enabled, you must interpolate 1/3/7/15 points by configuring AUDPWM_PWM_CFG.interp_rate.
- Although the left and right channels are masked, the whole processing logic of the Audio PWM is still running.
- The required clock frequency = the audio sample rate * the duty cycle resolution * (the interpolation rate + 1). For example, when the audio sample rate is 16KHz, the output width is 11bits, and interpolate 1 point, the frequency of sclk_audpwm = 16K * (2^11) * (1+1) = 65.536MHz. You should configuring the CRU to generate the accurate divided clock.

f